Anonymous Authentication

Andrew Y. Lindell[*]

Aladdin Knowledge Systems Inc.
Andrew.Lindell@aladdin.com

Abstract

In this paper, we study the seemingly paradoxical notion of *anonymous authentication*: authenticating yourself without revealing your identity. Before studying the technical aspects of anonymous Internet connections and anonymous authentication, we begin with a discussion of privacy and why it is important. Although the necessity of privacy is well accepted in the computer security community, the concrete reasons as to *why* are often not understood. We will demonstrate that such an understanding is crucial for providing technical solutions that solve the real problems at hand. Given this background, we briefly survey the known technologies for constructing anonymous Internet connections. Finally, we study how it is possible to achieve anonymous authentication (on top of anonymous Internet connections). We present definitions and a number of different protocols with different properties. In addition to the basic problem, we also show how it is possible (in a certain model) to achieve revocable anonymity, and explain why this is of importance.

1 Introduction

Internet chat rooms are a huge success. However, this success comes together with serious concern by parents about their children's activities on these sites. This is not mere paranoia: Detective Chief Superintendent Keith Akerman, head of the computer crime group of the Association of Chief Police Officers (of England, Wales and Northern Ireland) reported that *20 percent of children who use computer chat rooms have been approached over the Internet by pedophiles* [1]. This same statistic also holds in the United States. The following is a quote taken from [2]:

> *"Based on interviews with a nationally representative sample of 1,501 youth ages 10 to 17 who use the Internet regularly:*
>
> - *Approximately one in five received a sexual solicitation or approach over the Internet* **in the last year**.
> - *One in thirty-three received an* **aggressive** *sexual solicitation — a solicitor who asked to meet them somewhere; called them on the telephone; sent them regular mail, money, or gifts."*

What is the solution to this horrific state of affairs? Without a doubt, the solution lies mainly in education and with family. However, technological solutions can also help (although, as usual, cannot solve the problem all by themselves). For example, Verisign and AOL together with i-Safe (an organization that aims to educate children about safe online behavior) demonstrated a safe

[*] Additional affiliation: Dept. of Computer Science, Bar-Ilan University, Israel. email: lindell@cs.biu.ac.il

chat room that used strong authentication to ensure that only authorized children have access. This solution has its own problems (if we are overly confident that everyone in the chat room is OK, then a pedophile who manages to steal an authentication device or otherwise gain access will have a much easier time). Other suggestions include devices that ensure that the user is a child, and not an adult. Some even propose using biometric devices that check that the finger of the user matches that of a child and not an adult.

Our aim here is not to determine which method is best, nor what is likely to succeed. Rather, our aim is to point out that whatever the exact solution, it must contain some type of authentication. At first glance this seems fine. However, one of the features that children like most about chat rooms is their ability to be anonymous if they so choose. Asking them to authenticate takes away this anonymity (it is possible to not publicize their name after they login, but once it is clear that the site owners know who they are and can pass their details on, this can significantly inhibit children in their interactions). Thus, what we would really prefer is *anonymous authentication* whereby the chat room server can verify that some authorized user is now logging in, but it has no idea exactly which one. In this scenario, it may also be desired to have *revocable anonymity.* This is of importance in the case that one of the users is harassing others and some action must be taken. Since authentication is anonymous, there is no way of cancelling the user's account or finding out who they are. Thus, it is necessary to be able to revoke the anonymity of a user, when needed. In order to preserve the anonymity of others, this should only be possible under court order (and enforced through a secure protocol).

Another scenario where anonymous authentication may be desired is on professional Web sites and forums. Consider the case of a service where professionals can ask technical questions and search technical material. Tracking what different users ask and look at is an excellent source of information for those carrying out (legal) industrial espionage. If users must authenticate (because registration or even payment is required), then this situation becomes much worse. The site owners and possibly others can obtain valuable information about what their members are researching and/or developing. This information can be highly valuable and damaging. Once again, anonymous authentication would solve this problem.

This paper – organization. In this paper, we begin by studying the issue of privacy from a *concrete perspective*, meaning that we look at concrete damages that can arise from a loss of privacy. This is in contrast to the typical approach that considers only abstract issues and a general feeling of "invasion" when private information is publicized. We argue that a proper understanding of the real problems of privacy is crucial for constructing technical solutions that actually solve the problem.[1] Given this background, we proceed to build the infrastructure needed for anonymous authentication. First, we survey how it is possible to achieve anonymity over the Internet, without authentication. This is a necessary requirement for any protocol providing anonymity. Following this, we proceed to the main topic of this paper: anonymous authentication.

We begin by formally defining the requirements of an anonymous authentication protocol and present two approaches for solving it. The first is simple, uses basic public-key encryption, and is an extension of ideas presented in [18] (for solving a different problem). The second is a novel application of ring signatures [20] to the problem of anonymous authentication. Finally, we discuss how anonymity can be revoked, and how it is possible to obtain partial anonymity using only password-based technology.

[1] This observation is due to personal experience. Once, when working with colleagues, we devised a cryptographic solution to a specific problem of privacy-preserving data mining. However, after speaking to privacy experts we understood that our solution – although intuitively appealing – actually completely missed the point!

Related work. The problem of anonymous authentication is not new and has been studied in [16, 6, 21]. A more general problem of anonymous credentials has also achieved much attention; see [7] for just one example. However, most of this work has focused on the problem where anonymity revocation is required.[2] This is indeed a more interesting cryptographic setting. However, most of the solutions rely on protocols that cannot be implemented using standard smartcard operations. In contrast, the focus of our work is on solutions that can use standard encryption or RSA operations and thus can be easily deployed with today's infrastructure.

2 Privacy – A Concrete Perspective

Most people typically associate the loss of privacy with an abstract feeling of "invasion" or loss of control. This feeling of invasion is an uncomfortable one and it results from the mere fact that others know a lot about us. Interestingly, if we ask people *why* they are disturbed by the fact that their "private information" has become "public", or *why* the thought of "omnipresent surveillance" bothers them, they often don't have a concrete answer. Rather, what is most likely disturbing is simply the fact that their realm of solitude (using the words of [22]) has been invaded. In the words of Warren and Brandeis, their "right to be let alone" has been compromised [25].

Keeping this in mind, let us consider the following example. Assume that a user often reads the newspaper online, including following links to related sites of interest and (anonymously) responding to articles through the newspaper's "comment on this" feature. Furthermore, assume that all of her actions on the site over years of use are recorded to the smallest detail. Over time, the online newspaper actually compiles a comprehensive dossier of the user's interests and opinions, and possibly can derive her political opinions and financial status. Despite the above, if the user never registered to the site, and so never divulged her identity, this aggregation of information is unlikely to be disturbing.[3] We stress that here the user maintains a consistent virtual persona, possibly via a fixed pseudonym, and the market provides many incentives to do this (e.g., offering specials to returns customers and so on). Compare this to the case that all of this information is stored together with the user's name, address and social security number. This would be considered by most people to be a gross invasion of privacy. That is, many people feel that their right to be let alone is compromised when others hold private information that can be linked directly to them (or if they fear that this linking can happen at a later time). This was demonstrated in a Business Week/Harris Poll in the year 2000: 35 percent of the people that they polled were "not at all comfortable" with the profiling of their online actions, while this jumped to 81 percent when the profiled information was linked to the user's real identity [3]. (The study does not show how many of the 35 percent answered in this way because they feared future linkage. We conjecture that this concern is of some significance.) Although we do not belittle the importance of a person's solitude, we will demonstrate that in many ways, this is the least of our problems. That is, we will show that in today's digital age, the concrete dangers that arise due to our lack of privacy are very real and go far beyond our abstract right to be let alone.

We will also demonstrate the importance of understanding these concrete dangers in order to ensure that our privacy-preserving solutions actually help. For example, one of the common suggestions for protecting privacy is to use a *pseudonym*. In this way, the user's identity is hidden

[2] An exception is the work of [21] that does not consider revocability and whose solution has some high-level similarity to ours. Despite this, it is not clear how to securely instantiate their protocol, and known public-key encryption schemes like RSA cannot be used.

[3] This is not true if the user is concerned that her identity may at some time be revealed. However, here we are assuming that there is some magical guarantee that her identity can never be learned.

as long as it is not possible to link the user's pseudonym to her real identity. A valid criticism of using simple pseudonyms is that it may be hard – if not impossible – to guarantee that the link between the user and her pseudonym is never found. We will show that even if it were possible to guarantee (with 100 percent assurance) that the pseudonym is never linked to the user's real identity, the privacy concerns arising from the aggregation of user information are far from solved.

In the next two sections below, we will study two very different issues that are related to (the lack of) privacy. The first is focused mainly on issues of *free speech*, while the second considers the damage that comes out of the recording and aggregation of a user's *actions* (for example, their consumption behavior). These issues are to some extent very different, but also have much in common. Specifically, the understanding that a user's actions are an expression of herself leads to the conclusion that the detailed recording of these actions (usually without the user's knowledge) raises serious privacy concerns. Before proceeding, we remark that much of the discussion below is based on the works of [22] and [26], which we highly recommend reading.

2.1 The Realm of Solitude

We begin by discussing some concrete ramifications of a situation in which a person cannot act anonymously. That is, we begin with the simplest case whereby a person's privacy is only considered to be compromised if their information can be *linked* to their real identity. For the purpose of the discussion, assume that there is no reliable pseudonym solution, and a person's identity is fully known in all of her online actions. It is easy to see that such a situation can lead to self-censorship and inhibition. In general, although the freedom of speech is a constitutional right, it can be greatly inhibited if people fear that their may be ramifications to their speech. For example, in the McCarthy era, people were afraid to speak honestly about communism – despite the fact that they were supposedly protected by the constitution. In 1950, Senator Margaret Chase Smith delivered a speech against McCarthyism in which she counted some of the basic principles of Americanism: *the right to criticize; the right to hold unpopular beliefs; the right to protest; [and] the right of independent thought.* She went on to state that the current situation is such that people are afraid to exercise their freedom of speech. Although McCarthyism is well in the past, is it inconceivable that modern threats may result in a similar state of fear? Today, the citizens of most democratic countries feel free to criticize (and support) their government's policies, and this is crucial for democracy. However, there are many countries where this ideal is far from being realized. Furthermore, there is no guarantee that the "Western world" will continue to offer this freedom forever. For example, it is a real possibility that the threat of terrorism will usher in a new era of fear of everything Islamic (for example). In such a case, an individual's religious affiliation, or even just academic interest, can make them a target. Given the fact that this affiliation or interest is likely to express itself in a user's online actions, it may be very easy to locate "suspects" (much easier than in the days of McCarthy). It is not difficult to imagine the dangers lying behind the aggregation of information relating to a user's beliefs and interests. However, many other types of data are also collected. For just one example, a user's consumer activities – what products are purchased, when and where – are commonly collected and stored. Although the danger in this seems to be more vague, we will show below (in Section 2.2) that it is very real!

If the actions of Internet users are tracked and recorded (or even if there is just a possibility of this happening), then they may be afraid to voice "unpopular beliefs" or exhibit "independent thought." After all, this information may be held for all time and we don't have any real way of erasing it. We stress that the abstract uncomfortable feeling that we mentioned above is now translated into something that has a real effect on our *actions*. This is called the *chilling effect*, and is a very concern in this context. Consider the following three examples:

1. Many universities have forums in which students can hold discussions and express their opinions. Part of these discussions relate to courses and lecturers, and we encourage students to voice their opinions honestly (albeit respectfully). However, if postings are not anonymous, or if *there is a real concern that anonymity can be revoked*, then students may not criticize – even when justified – because they fear retaliation by the lecturer involved or the institution.

2. Many teenagers are heavy users of chat sites where they can freely express themselves. This freedom is often due to their belief that they are behaving anonymously and so do not need to fear ridicule from their peers if they voice unpopular beliefs. We are all familiar with the power of peer pressure. The Internet offers a place where teenagers can express themselves without this pressure.

3. The Internet is an important source of information. Sometimes the information being sought is highly sensitive and so people feel freer to turn to the Internet. For example, there are many Web sites that serve as a source of help to people in time of psychological crisis. Other important sources are for sensitive medical information, and information related to sexual health and practice. A teenager, for example, who searches for such information has an expectation of privacy (and otherwise, may not go to these sites). The expectation of anonymity here is crucial and the teenager clearly expects that his friends and/or parents cannot discover his searches without his consent. (Of course one would hope that parents can be a source of information here, but unfortunately this is not always the case.)

The examples above differ with respect to the source of fear (retaliation from a lecturer, ridicule by peers, or deep embarrassment). However, they are all real examples of where a lack of anonymity can limit the freedom of users. We remark that providing anonymity also has its dark side, because people may feel free to unjustifiably and maliciously slander others. However, this is also true of free speech which can and has been used to promote racism, for just one example. Despite this, the importance of free speech to modern society is such that we accept its dark side as well.

2.2 The Impact of Privacy Loss Without Identification

In the above, we have discussed the impact of privacy in a situation where a user's information can be linked to his or her real identity. Thus, in those cases it is the *identification* of the actual user that causes the loss of privacy and the damage. We will now show how privacy can be compromised even if the user's real identity is never revealed! This may come as a surprise. We present three examples, the first two of which are taken from [26]. The damage in each of the examples below is due to the fact that our actions are typically *linkable* (e.g., via our IP address or some other identifier). Thus, it is possible to process a user's past actions and potentially use that information to influence current decisions. Importantly, users are often not aware of the fact that their information is linked and are therefore placed at a significant disadvantage (this is sometimes called "privacy myopia" because people are unaware of how this might affect them in the future).

Price discrimination. Consider an online shopping scenario in which a user's shopping habits are recorded. By applying data mining techniques, these records can be used to improve customer experience by offering her products that are likely to be of interest. One example of the successful use of this technique is that of `Amazon.com` who offer products to customers, based on their previous purchases and searches (and on information gathered from other customers as well). Although such technology can be useful and positive, it can also be used for unfair price discrimination.[4] For example, if a user's profile shows that they do not "shop around" and usually buy as soon as they find what they are interested in, then the shopping site may charge the user higher prices. Now, price discrimination is often considered a positive economic force as it enables sellers to charge more to those willing to pay more and less to others. However, in our example there is an inherent asymmetry: the seller has a lot of information about the buyer and what they are willing to pay. In contrast, the buyer does not have equivalent information about what price the seller is willing to sell for. To make things worse, the buyer is not even aware that the seller has her purchase profile and typically assumes that she is being charged just like everyone else (this is exactly the privacy myopia mentioned above). This lack of symmetry between the buyer and seller creates an unfair disadvantage to the buyer. Stated differently, this shows that the availability of personal information allow for the unfair transfer of consumer surplus to the vendor. At this point it is crucial to notice that this holds even if the buyer's true identity is unknown at the time the price is offered. The only information that the seller needs is a link (say, by IP address) between the current customer and her profile. Needless to say, using a consistent pseudonym would provide no protection here whatsoever.

Manipulation and personal autonomy. We began our discussion of privacy with an example of an online newspaper. We discussed the fact that if the user's identity *cannot* be revealed, then it is more likely that they will not view the recording of their use history as an invasion of privacy. As above, we claim that significant damage can be incurred by the user even if her identity is never revealed. In particular, as in the above example, the newspaper may use the user profile in order to tailor the presentation directly to that user. Although this can be positive (e.g., first presenting articles that are likely to be of interest), it can also be used for negative means. For example, it is possible for the newspaper to target the individual user and modify content in order to influence them unfairly. Of course, newspapers are often biased. However, this is generally known and the bias is equal for all users. The ability to individually target users and modify content to exert influence enables the newspaper to manipulate users against their will. This is an infringement on their personal autonomy and is a serious concern. Of course, this example extends to all online content providers. Once again, this potential manipulation can take place without ever knowing the identity of the user; all that is needed is the ability to link a user's past and present actions. To a certain extent, this is what some online ad firms try to do through cookies and other surveillance methods.

Industrial espionage. Today, there are many forums that are used by software developers to ask each other questions and discuss development issues. These forums are a helpful tool, but also a source for industrial espionage. This is due to the fact that the history of a user's questions can often point to the problem that she is trying to solve. This can therefore reveal the new feature or product that a competitor is working on. A naive solution to this problem is to have the developer ask her questions under a pseudonym that will not reveal the company that they are working for.

[4] Another problem that arises in this context is that of *incorrect information.* A friend of mine complained to me that she is always offered books on cryptography by Amazon because she once bought me a few books on the subject. She has no interest in cryptography, but has no way of correcting the situation. This is a relatively benign example. However, people who are mistakenly labeled as being high-risk to insure (e.g., labeled as having heart disease or participating in dangerous sports) can find themselves charged higher rates, even though the labeling is incorrect. This issue is closely related to privacy and the need for anonymity, but we will not elaborate further on it.

However, the mere fact that there *exists* a company who is developing some feature can be enough information. Thus, once again, the actual identity of the user does not need to be learned in order for damage to occur.

The conclusion from the above is amazing – it is possible to use someone's information against them, without ever having the link between the virtual persona and the real one. In all of the examples above, the damage is due to *data aggregation* and the possibility of linking a user's prior actions to her current ones. This suggests that in order to preserve our privacy, we must look for solutions that prevent anyone from linking our different actions.

At the risk of repeating ourselves, we stress that understanding the above dangers is crucial for constructing effective privacy-preserving solutions. In particular, most users would probably feel comfortable with a solution where they use a single pseudonym for all of their online shopping, as long as it is guaranteed that it can never be linked to them. However, as we have shown, this pseudonym solution does not solve the real problems that arise out of the lack of privacy (at least, not all of the problems).

3 Anonymous Internet Connections

3.1 The Security Goal

Many of the privacy problems that arise out of Internet use can be solved by using a mechanism that achieves anonymous Internet connections. An important feature of these mechanisms is that a user's actions are unlinkable. This means that, as far as a server, router or external eavesdropper is concerned, it is impossible to distinguish the case that ten page requests from a given Web site originated from a single user from the case that they originated from ten different users. In reality, this goal as stated cannot be achieved. To go to the extreme, if only one person is using the Web at a given time, it is clear that this user is the one making all page requests. Fortunately, in practice, this is not the case and significant guarantees of anonymity can be achieved.

We distinguish between two different tasks:

1. *User anonymity:* This task involves modifying regular packets so that the user's requests contain no identifying information. This is not easy if the server requires no authentication, and very difficult if it does. This latter case is dealt with in Section 4 below; here we will assume that the user communicates with a Web server without any registration or other process.

2. *Unlinkability:* This task involves ensuring that an attacker who can view Internet traffic is unable to trace which servers a given user is communicating with.

We stress that the above two tasks are different and *incomparable*. In other words, a solution to each one does not imply a solution of the other. Thus, any full solution to the problem of anonymity must provide complementary tools for achieving both tasks. We demonstrate the incomparability of these two tasks by showing how each can be achieved without solving the other.

It is possible to solve user anonymity by just stripping all identifying information from the traffic being sent. However, an attacker carrying out traffic analysis can easily see who the user is communicating with by following the packets from their origin to their destination. For the other direction, assume for a moment that we have a magical mechanism that can hide all Internet traffic (and so no tracing is possible). Then, assume that a user connects to a server using this mechanism but without stripping off its identifying information (assume, for example, that the packets are encrypted with a key known only to the user and server and so this identifying information is not available to the attacker carrying out traffic analysis). In this case, even though by magic no traffic analysis is possible, there is clearly no user anonymity. Despite what we have said above, it is possible to unify these two tasks into one by saying that anonymity is preserved even in the presence of an attacker who controls the server *and* can carry out traffic analysis. Having said this, our focus will be on solving unlinkability; the problem of user anonymity can be solved without too much difficulty by a user itself. From here on, we will refer to a method that achieves the goal of *unlinkability* as a mechanism for anonymous routing.

Defining anonymous routing. Before attempting to solve the problem of anonymous routing, it is crucial that a precise definition of security be formalized. Unfortunately, the literature is abound with different notions, some better and some worse. Furthermore, they are highly technical. Since our main focus in this paper is *anonymous authentication* (see Section 4), we will limit ourselves in this section to a survey of techniques, together with intuitive explanations as to why (at least some degree of) anonymity is achieved.

Additional issues. We stress that there are a number of other highly important issues that arise when designing a system for anonymous routing. For example, how vulnerable is the system to a denial of service? In many cases there is a tradeoff between anonymity and robustness. That is, a stronger guarantee of anonymity often comes together with a system in which it is easy to block communication, and vice versa. Nevertheless, we ignore these other issues here and focus on how anonymity can be achieved.

3.2 Anonymous Routing Mechanisms – An Overview

There are a number of different approaches and variants to achieving anonymous routing. We will briefly describe a few of them, starting with mix-nets that was the first method suggested.

Mix nets. This approach was proposed by Chaum [8]. The idea is that users send their network packets via a computer called a *mix*. This computer receives a number of different packets and randomly permutes them before forwarding them on. Of course, there must be some transformation carried out on each packet in order to prevent tracing a packet from its source to destination. Naturally, this transformation is that of encryption.

In more detail, assume that a series of users $U_1, \ldots, U_n$ wish to send respective messages $x_1, \ldots, x_n$ to respective servers $S_1, \ldots, S_n$. We assume that each server has a public-key and we denote these keys by $pk_1, \ldots, pk_n$. The users use the mix machine, denoted $\mathcal{M}$, who has a public-key $pk_{\mathcal{M}}$. Now, each user U_i sends the ciphertext

$$c_i = E_{pk_{\mathcal{M}}}(S_i, E_{pk_i}(x_i)).$$

to the mix machine $\mathcal{M}$. Note that we denote encryption of a message m with public-key pk by $E_{pk}(m)$. Thus, the ciphertext above is generated by the user U_i by first encrypting x_i under the public-key of the server S_i and then re-encrypting this ciphertext, together with the identity of the server (for simplicity we don't differentiate between the server's address and identity), under the mix's public-key.

Upon receiving the $c_1, \ldots, c_n$, the mix machine $\mathcal{M}$ chooses a random permutation π over the indices $\{1, \ldots, n\}$. Then, $\mathcal{M}$ decrypts all of the ciphertexts and sends the pairs $(S_i, E_{pk_i}(x_i))$ in the permuted order. Denoting $\pi(i)$ the result of the permutation π applied to i, we have that $\mathcal{M}$ first sends $E_{pk_{\pi(1)}}(x_{\pi(1)})$ to server $S_{\pi(1)}$, then $E_{pk_{\pi(2)}}(x_{\pi(2)})$ to server $S_{\pi(2)}$, and so on until it finally sends $E_{pk_{\pi(n)}}(x_{\pi(n)})$ to server $S_{\pi(n)}$. We remark that if the servers have to reply to the users, as in the case of Web surfing, the mix machine $\mathcal{M}$ can send the pair

$$\langle E_{pk_{\pi(i)}}(x_{\pi(i)}), E_{pk_{\mathcal{M}}}(U_{\pi(i)})\rangle$$

to $S_{\pi(i)}$. Then, $S_{\pi(i)}$ includes the ciphertext $E_{pk_{\mathcal{M}}}(U_{\pi(i)})$ in its reply message. The mix machine $\mathcal{M}$ can decrypt this ciphertext, and then knows that the message is intended for $U_{\pi(i)}$ (note that the identity of the user is encrypted under the mix's public-key so only it can decrypt). Of course, these reply messages must also be sent using a mix.

The above method has the drawback of relying on a single (trusted) mix server. This can be improved by using a series (or cascade) of mix machines. In this way it suffices that just a single mix machine is not corrupted, and anonymity is guaranteed. A series of mix servers carrying out these operations is called a mix net.

Although intuitively appealing, it is important to note that the level of anonymity for this method depends heavily on the real Internet traffic at any given time. In particular, the timing and number of packets sent can reveal the pairing between users and servers. Consider first the case that a user and server are active at the same time interval, and dormant before and after this interval. In such a case, it is likely that they were communicating with each other and this can be detected. Likewise, if the number of packets sent between a given user/server pair is different from other active users and servers, then this can be enough to detect that they are interacting with each other; see [15, 12, 27]. Thus, anonymity is preserved in a mix if the following two assumptions hold:

1. At least one of the mix machines in the cascade is honest, and

2. The users and servers in the network send and receive almost the same amount of traffic.

Although the first assumption is a reasonable one, the second may be problematic. Nevertheless, this is a somewhat inherent problem in traffic analysis and it seems that it can only be solved by mandating that users and servers regulate their flow of traffic so that it is similar to others.

Onion routing. This is a variant of a mix-net in which the series of mix machines is chosen at random at the time that a connection is made between a user and server. There are a number of advantages of this method, and we refer to [14, 24] and reference therein.

Dining cryptographers. This approach, also proposed by Chaum [9], enables a set of parties to broadcast messages so that it is impossible to know which party broadcasted which message. We will describe the solution for two parties, Alice and Bob. Assume that Alice and Bob wish to broadcast respective messages x_A and x_B, and that they share two secret random strings k_1 and k_2 and a secret random bit b. Then, they broadcast messages as follows:

1. If $b = 0$, Alice broadcasts the pair $\langle k_1 \oplus x_A, k_2\rangle$ and Bob broadcasts the pair $\langle k_1, k_2 \oplus x_B\rangle$.

2. If $b = 1$, Alice broadcasts the pair $\langle k_1, k_2 \oplus x_A\rangle$ and Bob broadcasts the pair $\langle k_1 \oplus x_B, k_2\rangle$.

Denote the pairs broadcast by Alice and Bob by $\langle c_1^A, c_2^A\rangle$ and $\langle c_1^B, c_2^B\rangle$, respectively. Then, observe that for any value of b, we have that the set $\{c_1^A \oplus c_1^B, c_2^A \oplus c_2^B\}$ equals the set $\{x_A, x_B\}$. In more detail, if $b = 0$ then $\langle c_1^A \oplus c_1^B, c_2^A \oplus c_2^B\rangle = \langle x_A, x_B\rangle$, whereas if $b = 1$ then $\langle c_1^A \oplus c_1^B, c_2^A \oplus c_2^B\rangle = \langle x_B, x_A\rangle$.

This implies that anyone seeing the messages broadcast by Alice and Bob can easily derive x_A and x_B. However, as long as k_1, k_2 and b remain secret, there is no way that any adversary can learn which party broadcast which message. Thus, anonymity is unconditional! Another advantage of this method is that once the parties have the initial keys and bit, there is no interaction required. This method can be extended to the multiparty setting, but there are a number of difficulties with it that have made it less popular than mix-nets.

Crowds. A completely different approach to achieving anonymity was proposed by Reiter and Rubin [19]. Their approach, as the name suggests, is based on achieving anonymity by blending in with a crowd. Informally speaking, users form crowds before any transactions begin. Then, instead of a user sending a message directly to a server, it sends it to a random member of the crowd who either forwards it to another random member of the crowd or sends it to the server (and this decision of what to do is also at random). This method guarantees a degree of anonymity, in that the server cannot know which member of the crowd sent the message. A rigorous analysis that considers the level of anonymity achieved for different adversarial attacks is provided in [19].

Our description above is very brief and ignores a large body of literature that deals with many problems and issues that arise when attempting to achieve anonymous routing. Nevertheless, our aim here was merely to provide intuition about how anonymity is achieved. In practice, there exist implementations that enable anonymous Web surfing and they seem to work well. (Having said this, there is still more research to be done in this area and the problem is far from being fully solved.)

4 Anonymous Authentication

Assume now that you have a perfect anonymous routing mechanism that provides both user anonymity and unlinkability. However, you now need to carry out a task that requires that you first *authenticate*! For example, you may wish to browse a document library that requires registration or even payment (and you don't wish to re-register or buy a new subscription every time you enter the site). In this case, there is no way that you can have user anonymity: if you are anonymous, then how can the server check that you are authorized to enter.[5] For this reason, anonymous authentication sounds like a contradiction in terms. In this section, we will show that it is actually possible to achieve, and even quite efficiently.

4.1 Defining Security

We begin by defining the network model in which an attacker works. Our first assumption, mentioned above, is that the anonymous authentication protocol works on top of an *anonymous routing method* that guarantees both user anonymity and unlinkability. Thus, we can assume that the user and server are connected by a magical communication channel that reveals nothing about the user's identity to the server or to any network adversary. This means that we can formally model the interaction between the user and server as *direct*, but where the messages received by the server reveal nothing about the user's identity. We remark that this approach does not take into account the possibility of a *man-in-the-middle attack* on the protocol. Nevertheless, since the server is not

[5] In contrast, there is no problem achieving unlinkability and this is of help as long as the server does not *store* any information about you. However, this is rarely the case (for example, logs are typically kept for security and other reasons).

anonymous, it is always possible to first set up a server-authenticated SSL channel between the user and server, and then run the protocol. By using message authentication, this ensures that the communication between the user and server during the protocol execution is not tampered with by any other adversary.

There are two security requirements on an anonymous authentication protocol:

1. *Secure authentication:* No unauthorized user should be able to fool the server into granting it access (except with very small probability).

2. *Anonymity:* The server should not know which user it is interacting with.

These two requirements seem to contradict each other (it seems that the server needs to know who the user is in order to ensure that only authorized users are granted access). Nevertheless, this can be reconciled by saying that a server learns that it is interacting with a user that belongs to a *defined set of authorized users*, but nothing more about which user it is in that set. We remark that this approach also enables the use of anonymous authentication even in applications where not all users have the same permissions. In such a case, the defined set of authorized users for any given user is exactly those users with the same permissions.

In this paper, we present two definitions of security. The first is a straightforward implementation of the above intuitive discussion. We say that a protocol that achieves the aforemention two requirement is a secure protocol for anonymous authentication. The second definition that we present relaxes the requirements and allows a malicious server to detect the identity of the user, at the price of this cheating being (always) detected by the user. That is, the anonymity of the user is only guaranteed if it does not detect cheating by the server. We call this verifiable anonymity and claim that in most cases it suffices. This is because if the user detects cheating by the server, it can disconnect before carrying out any operations. Thus, essentially nothing is learned by the server (except that the user tried to connect). Furthermore, if this happens, the server's malicious behavior is revealed and so the user knows that it cannot be trusted. Due to their technical nature, we present the formal definitions in Appendices A.1 and A.2.

4.2 Constructing Anonymous Authentication Protocols

In this section we present protocols that achieve verifiable and full anonymity. We begin with verifiable anonymity because it is much easier to achieve.

4.2.1 A Protocol for Verifiable Anonymity

The protocol that we present here is based on [18] but is far simpler and more efficient. In order to motivate the protocol, and obtain some intuition about how anonymity can be achieved, assume that the authorized set of users are all given the same password or secret key. In this case, the server can never know which user is authenticating, because all users have the same secret information. This would provide perfect anonymity. However, it is clearly unacceptable because revoking the permissions of one user would require changing everyone's key or password. The idea that we will use is to ensure the same behavior by every user, irrespective of its identity. Informally speaking, the protocol works by the server first encrypting a random challenge w under all of the user's public keys. The user then decrypts the ciphertext associated with its private key and obtains the plaintext w. The user then returns w to the server, proving its ability to decrypt. Since only authorized users can decrypt, this in turn proves that the user is authorized. If the server follows the protocol, this provides perfect anonymity (because all users would obtain the same w). However, if the server cheats it can learn the identity of the user. Specifically, consider a server who chooses ℓ different random challenges $w_1, \ldots, w_\ell$ and encrypts w_i with the public-key pk_i. In this case, user U_i returns w_i and fully reveals its identity. We solve this problem by forcing the server to *prove* that it encrypted the same w under all public-keys. One possible way of implementing this is to have the server prove before the user replies. However, this is expensive and involves a significant computational cost. We therefore have the server prove that it acted appropriately *after* the user replies. We note that if the server indeed cheated as above, then it will detect the identity of the user. This protocol therefore does not achieve full anonymity. Nevertheless, since the user will detect such cheating, the protocol does achieve verifiable anonymity.

In the formal protocol definition we use the following notation. The encryption of a string w using public-key pk and random coins r is denoted $c = E_{pk}(w; r)$. Recall that any secure public-key encryption scheme must be probabilistic and so random coins must be involved in the encryption process. We remark that given w and r it is possible to verify that $c = E_{pk}(w; r)$ by just running the encryption process again from scratch. By unambiguity of decryption, it holds that for every c there is just one w for which there exists an r such that $c = E_{pk}(w; r)$.[6] Finally, we denote the decryption of a ciphertext c using private key sk is denoted $w = D_{sk}(c)$. We are now ready to present the protocol.

Protocol 1

- **Input:** *The user U_i and server S both have ℓ public keys $pk_1, \ldots, pk_\ell$ for an encryption scheme, and U_i has the private-key sk_i associated with pk_i, for some $1 \leq i \leq \ell$.*

- **The protocol:**

  1. *The server S chooses a random string w of length n and random coins $r_1, \ldots, r_\ell$ such that each r_j is of the length needed to serve as randomness for encrypting w under E.*
     For every $j = 1, \ldots, \ell$, the server S computes $c_j = E_{pk_j}(w; r_j)$.
     S sends $c_1, \ldots, c_\ell$ to the user U_i.
  2. *Upon receiving $c_1, \ldots, c_\ell$, the user U_i computes $w = D_{sk_i}(c_i)$ and sends w back to S.*
  3. *Upon receiving a string w' from the user, the server S grants access if and only if $w' = w$. If access is granted, S sends $r_1, \ldots, r_\ell$ back to U_i.*
  4. *User U_i verifies that for every $j = 1, \ldots, \ell$ it holds that $c_j = E_{pk_j}(w; r_j)$. If not, it outputs cheat_S and halts. Otherwise, it is granted access and continues.*

Note that in Step 4 of Protocol 1, the user U_i is able to verify that the server indeed encrypted the same string w under all the public keys. Thus, either U_i is guaranteed that its anonymity is perfectly preserved, or it catches the server cheating. This is exactly the property of verifiable anonymity described above. We have the following theorem, that is *formally proven* in Appendix B:

Theorem 2 *Assuming that the public-key scheme used is secure again chosen-plaintext attacks, Protocol 1 is a secure protocol for verifiable anonymous authentication.*

[6] This actually assumes that decryption always works correctly. However, in our application the public keys are honestly chosen by the users and so this holds with overwhelming probability.

We now discuss some properties of our protocol.

Efficiency. The server and user both need to compute ℓ encryptions: the server in order to compute $c_1, \ldots, c_\ell$ to send to the user, and the user in the last verification step. In addition, the user needs to carry out one decryption operation. Importantly this means that only one operation using a private key is needed by the user (and this is a regular decryption operation). Assuming that users use smartcards – as is necessary for obtaining strong authentication – we have that the smartcard is used to carry out one decryption only. The rest of the work by the server and user is carried out on a regular machine, making the protocol efficient for reasonable values of ℓ.

Implementation issues. Protocol 1 is very appealing in its simplicity. However, there are a number of issues that we have somewhat "swept under the carpet" if it is to be actually implemented. In particular, the protocol makes two implicit assumptions: (1) the set of authorized users is known, and (2) the user knows the public keys (or certificates) of this set of authorized users. In practice these issues are non-trivial to solve. For example, if the set of authorized users is huge (say hundreds of thousands), then it will be infeasible to use the entire set. Rather, in practice some subset must be chosen and thus known to (and agreed upon by) both the user and server. Since the user must always be in this subset (without revealing to the server its identity), we propose that the *user choose the subset of users to be used in the protocol.* This yields the property that the user knows that it is hiding amongst ℓ random users. Given a reasonable ℓ (say $\ell \approx 100$), this yields strong privacy guarantees. (It is somewhat related to k-anonymity [23] but the guarantees here are much stronger, because the user is provably hidden amongst ℓ *random users.*)

Assume now, as above, that the user chooses which subset of users is in the set. How does the user obtain all of the public keys or certificates of the users in the subset? One possibility is for the server to send them to the user. A solution like this would have the server publish the certificates of all the users of the Web site. Then, a user would choose a random subset, including herself, and download those certificates. However, there is one significant problem with this solution: A malicious server may publish a majority of fake certificates that do not correspond to any real user. For example, assume that 90 percent of the certificates are fake. In this case, a user choosing a random subset of size ℓ will use a set in which only approximately $\ell/10$ of the users are real. Thus, its anonymity is seriously degraded. (Note that the server knows the identity of the set of users being used and knows that the fake certificates are not real users. Thus, the user is really only hiding among $\ell/10$ legitimate users.) The only solution that we see to this problem is for the user to obtain the certificates or public keys from other users themselves, or to somehow validate that they correspond to real users in a different way (e.g., by contacting the users directly).

We stress that when the user chooses the subset of users, the server must verify that all of the certificates sent correspond with authorized users. Furthermore, if there are different permissions to the different users, then the lowest common level of permissions must be provided.

Privacy ramifications. We remark that our protocol assumes that the identities of the authorized users are known to all. In reality, this may not be the case. However, this is inherent to the problem and it means that anonymous authentication can only really be used when the identities of registered users is not secret. One partial measures that may be of some help is to register users that never use the site (and so no user is fully implicated by being on the list). We remark that a solution by which only a user's public-key is revealed and not their identity and/or certificate doesn't really help because the public-key may be linked to the identity elsewhere. Furthermore, this reawakens the problem of how other users know that the keys is real.

4.2.2 Achieving (Full) Anonymity

In this section, we present a completely different approach to anonymous authentication that has two advantages. First, it achieves full (rather than verifiable) anonymity. Second, the bandwidth of the protocol is a single signature (e.g., 2048 bits when 2048-bit RSA is used). The computation of both the user and server is the same as in Protocol 1; that is, ℓ public-key operations each and one private-key operation for the user.

Background – ring signatures. Ring signatures, proposed by [20], are a way of signing on a message so that it is guaranteed that the signer belongs to some specified set of users, but it is impossible to know which user actually signed. We stress that unlike group signatures [10], there is no central authority or fixed group. Rather, a signer can, in an ad hoc manner, choose a set of parties and incorporate them in a ring (without even asking their permission). Ring signatures have a number of applications, one particularly interesting one is leaking secrets in a semi-anonymous way. For example, consider a member of Congress who wishes to leak a some secret information to the press. That member of Congress wants to be sure that no one can detect who actually leaked the information. However, she also wants to be sure that anyone seeing the information is convinced that it was provided by *some* member of Congress. This can be achieved by the member signing on the information with a ring signature, where the ring includes all members of Congress. In this way, only a member of Congress could have signed, but no one knows which one actually did!

We propose a novel use of ring signatures for the purpose of anonymous authentication. Before showing how this is done, we discuss the notion of ring signatures in greater detail. The security of ring signatures is defined as one would expect. Specifically, a signature σ on a message m is associated with a ring of public keys $pk_1, \ldots, pk_\ell$. Then, unforgeability is formalized by saying that no adversary without knowledge of any of $sk_1, \ldots, sk_\ell$ can output a pair (m, σ) where σ is a valid ring signatures with respect to the ring of keys. Of course, as usual, the adversary is given access to a signing oracle and succeeds if it outputs a valid signature on any message not queried to its oracle. In this sense, unforgeability is essentially the same as for regular signatures. Regarding anonymity, the requirement is that for any signature σ generated with private key sk_i, the probability that the signing algorithm with sk_i outputs σ equals the probability that the signing algorithm with sk_j outputs σ, for all $j \neq i$. This is *perfect anonymity*; computational anonymity can also be defined.

A number of protocols have been proposed for ring signatures since its invention. However, a significant advantage of the original scheme of [20] is that – as with our protocol – it requires a single standard RSA secret operation by the signer (that can be carried out on a smartcard) and $\ell - 1$ public operations that can be carried out on a regular computer. We refer the reader to [20] for details.

Anonymous authentication using ring signatures. A ring signature scheme automatically yields an anonymous authentication protocol. One possibility is simply for the server to send a random challenge w and to grant access if the user returns a *ring signature* on the random challenge. Anonymity is provided by the "signer ambiguity" of the scheme, and the fact that it is a secure authentication protocol follows by the unforgeability of the signature scheme.

Another way of using ring signatures for anonymous authentication is for the user to connect to the server using *SSL with server and user authentication*. In this case, the user authenticates by signing on a hash of the handshake messages. All we need to do now is to replace this standard digital signature with a ring signature instead. Specifically, instead of sending a single user certificate as part of SSL, the user sends a set of ℓ certificates. Then, it produces a ring signature on a hash of the handshake messages, where the ring is formed from those ℓ certificates. When the server successfully verifies the ring signature, it grants access to the user if and only if all of the certificates in the ring belong to authorized users. (As mentioned above, if the users have different permissions, then the lowest common access is what is granted.) It is straightforward to verify that this results in a secure anonymous authentication protocol.

Ring signatures vs. Protocol 1 – a discussion. On the one hand, the ring signature protocol is advantageous in that in can be directly incorporated into SSL and does not require any additional communication. However, the known practical ring signature protocols rely on somewhat problematic assumptions. For example, the protocol of [20] that uses RSA relies on an ideal cipher. The problem is that modern block ciphers (like 3DES and AES) don't necessarily live up to the expectations of such an ideal cipher (see [17] for an example). In contrast, Protocol 1 relies only on standard CPA-secure public-key encryptions and does not assume random oracles or ideal ciphers. This provides it with an advantage regarding its security.

4.3 Revocable Anonymity

In some cases, although anonymity is highly desired, it is important to be able to revoke anonymity. Take for example the use of anonymous authentication in order to protect chat rooms from pedophiles and other unwanted members. What can be done if one of the users behaves highly inappropriately (or a pedophile stole an authentication device and is using it to gain access)? It is impossible to cancel their account and access because we don't know who the user is! In this case, we would like to be able to obtain a court order allowing us to *revoke* the anonymity of this user. There exist solutions for revocable anonymity (e.g., [16, 6, 7]). However, these do not use standard certificates and smartcard operations and thus are more difficult to implement and deploy. Unfortunately, we do not currently have a good solution to this problem that can use standard smartcard operations. However, if we are already willing to construct a special-purpose device (using Javacards, this is not too difficult today), then we have the following very simple solution.

Let pk_C be the public encryption key of a *court authority* who is authorized to revoke anonymity (of course, this public-key should be provided in a digital certificate, ensuring all users that it indeed belongs to the court). Then, in addition to carrying out the protocol for anonymous authentication, the user includes an encryption of its identity. If needed at a later time, the court can decrypt the ciphertext and obtain the user's identity.

So far, this sounds like it can be done without a special-purpose device. However, the problem is that malicious users are unlikely to behave properly. Thus, they may send an encryption of garbage or of someone else's identity. We remark that since the authenticating server cannot decrypt, it is unable to check that the encryption is valid.

Consider now the case that users are provided with authentication devices (smartcards) by the organization who authenticates them. This organization can then ensure that the smartcards that carry out the authentication will only work in the above way. Essentially, this means that users cannot behave in a different way: they need the smartcard to authenticate, but then the smartcard also passes on their identity in an escrowed way. Since the authenticating organization distributes the smartcards, it also initializes the keys. This prevents replacement of the smartcard with a different one or with software. This solution still needs care because if the encryption of the identity is not somehow bound to the transcript, the user can replace the encryption generated by the smartcard with an encryption to garbage and, once again, the authenticating server will not be able to detect this. We therefore have the following:

Revocable anonymous authentication protocol. Our first protocol uses ring signatures. In such a case, when the user signs on the SSL handshake messages (as described above), it includes an encryption of its identity under the court authority's public key. The server then verifies the ring signature as before, but includes the ciphertext as part of the signed message. Note that if a user has control over its secret key, then it can include an encryption to garbage. It can also implicate another user by encrypting someone else's identity. However, given that only the smartcard can compute a ring signature – and it always includes an encryption of the identity in that signature – the user is unable to prevent the court authority from revoking its anonymity. We stress that the user is unable to separate the encryption of its identity from its ability to authenticate because the server verifies the signature on the SSL handshake messages together with the ciphertext (encryption of its identity) generated by the smartcard.

Revocable anonymity based on Protocol 1. The above methodology for including revocability in the ring-signature method does not translate to Protocol 1. The reason is that the user just returns w which does not seem to be bindable to an encryption of its identity under the court authority's public-key. Recall that sending such an encryption separately doesn't help because the user can modify what it receives from the smartcard before sending it on. We therefore present a different solution which uses *CCA2-secure* or *non-malleable* encryption (see below).

The idea behind the protocol is that, as before, the server sends an encryption of the same n-bit string w under each public-key pk_i. The smartcard then decrypts the i^{th} ciphertext and computes a reply $E_{pk_S}(w \parallel E_{pk_C}(i))$, where pk_S is the encryption key of the server, pk_C is the encryption key of the court authority, and $\parallel$ denotes concatenation. The server then decrypts and grants access if the first item is w; the second item is stored for revoking anonymity later, if necessary. Now, assuming that a malicious user cannot tamper with the reply ciphertext, it cannot modify the encryption so that its identity (encrypted under pk_C) is modified. Of course, a user can always modify a ciphertext, so what do we mean here? The idea is that of *non-malleability of encryptions* [13] and it means that it is impossible for anyone not knowing the secret key to modify the ciphertext so that the result is an encryption of a *related* plaintext. More concretely, given $c = E_{pk_S}(w \parallel E_{pk_C}(i))$ it is infeasible to generate a ciphertext c' which encrypts the same w and anything else, unless you know w or you know the secret-key sk_S. However, the malicious user knows neither and thus it can either modify c and fail to gain access (because when the server decrypts it will obtain something that will not begin with w), or it can leave c as is, in which case its anonymity can be revoked if necessary. It has been shown that any CCA2-secure encryption scheme is non-malleable.[7] Thus, it is possible to use the Cramer-Shoup encryption scheme [11], or OAEP [5] (now standardized in PKCS#1, v2.1) if one is happy to work in the random oracle model.

The detailed protocol follows. In the protocol, we refer to a smartcard SC and a user U rather than just to a user, to stress that the "smartcard operations" are carried out atomically on the smartcard. We also note that E^{CPA} refers to a (standard) CPA-secure public-key encryption scheme, whereas E^{CCA} refers to a CCA2-secure public-key encryption scheme.

[7] A CCA2-secure encryption scheme is one that remains secure even if the adversary has access to a decryption oracle that it can use to decrypt any ciphertext, except the one that it is trying to learn information about.

Protocol 3

- **Input:** *The user smartcard SC_i and server S both have ℓ public keys $pk_1, \ldots, pk_\ell$ for an encryption scheme, and SC_i has the private-key sk_i associated with pk_i, for some $1 \leq i \leq \ell$. In addition, the smartcard has the public-key pk_S of the server and the public-key pk_C of the court authority, and the server has the private-key sk_S associated with pk_S.*

- **The protocol:**

  1. *The server S chooses a random string w of length n and random coins $r_1, \ldots, r_\ell$ such that each r_j is of the length needed to serve as randomness for encrypting w under E.*
     For every $j = 1, \ldots, \ell$, the server S computes $c_j = E^{\text{CPA}}_{pk_j}(w; r_j)$.
     S sends $c_1, \ldots, c_\ell$ to the user's smartcard SC_i.
  2. *Upon receiving $c_1, \ldots, c_\ell$, the smartcard SC_i computes $w = D^{\text{CPA}}_{sk_i}(c_i)$. Next, it computes $c = E^{\text{CCA}}_{pk_S}(w \parallel E^{\text{CPA}}_{pk_C}(i))$, and sends c back to S.*
  3. *Upon receiving a ciphertext c from the user, the server S grants access if and only if the first n bits of $D^{\text{CCA}}_{pk_S}(c)$ equals w. If access is granted, S sends $r_1, \ldots, r_\ell$ back to the user U_i.*
  4. *User U_i verifies that for every $j = 1, \ldots, \ell$ it holds that $c_j = E^{\text{CPA}}_{pk_j}(w; r_j)$. If not, it outputs cheat_S and halts. Otherwise, it is granted access and continues. (Note that the user checks this and not the smartcard.)*

We stress that there must be a way of verifying that the key pk_C used by the smartcard actually belongs to the court authority and not to the server (or anyone else). This can be achieved by making this key or certificate readable but not modifiable by the user. Thus, a user can check that the smartcard will use the correct court authority's public-key, but cannot modify it to prevent revocation.

There is one subtlety that we must remark upon here: the user's anonymity in this protocol relies on the assumption that the smartcard provided by the authenticating organization works correctly. That is, since the user cannot use its own software, it must rely on the authenticating organization that it did not change something in the protocol in order to be able to identify the user.

Security of the protocol. The fact that Protocol 3 is a secure authentication scheme, and achieves computational anonymity can be proven in an almost identical way as for Protocol 1; see Appendix B for a full proof. Regarding revocability, we must first discuss how to model security. The basic idea of this model is that a malicious user must either choose to not obtain access, or if it does gain access, it cannot prevent its anonymity from being revocable. Thus, we model the user as a *man-in-the-middle* adversary that sits between the smartcard and the server. This adversary's aim is to successfully authenticate itself to the server, without the server receiving $E_{pk_C}(i)$. Now, notice that the only thing that the man-in-the-middle adversary sees is a series of encryptions $c_1, \ldots, c_\ell$ and the ciphertext c. Security is proven by showing that any adversary succeeding in its task can break the CCA2-security (or non-malleability) of the encryption scheme.

An extension to full anonymity. If CCA-secure encryption is already being used, it is possible to modify Protocol 3 so that full anonymity and revocability are achieved. This can be done using ideas from the protocol of [18] for deniable ring authentication.

4.4 Password-Based Anonymous Authentication

A natural question that arises after seeing the above protocol is: can anonymous authentication be achieved without using a public-key infrastructure and/or smartcards? Stated differently, can it be based on passwords, one-time passwords and the like? On the one hand, the above protocols are inherently "public key"; on the other hand, this somewhat limits their applicability. In this section we present a partial solution to this problem, but stress that the level of security achieved is less than what we have seen until now.

We will demonstrate the solution here with respect to *any* standard authentication mechanism. This can be later implemented with passwords, one-time passwords, biometrics or anything else. The high-level idea is as follows:

- *Step 1 – standard authentication:* In this step, the user uses the given standard authentication protocol to authenticate itself to the server.
- *Step 2 – register temporary public key:* After the user authenticates itself and the server grants access, the user generates a public/private key pair (pk, sk) and sends pk to the server. (Note that the temporary key generation takes place on the user's machine and is therefore not prohibitively expensive.)
- *Step 3 – disconnect and reconnect using anonymous authentication:* At this point, the user disconnects its connection and reconnects using an anonymous authentication protocol and its temporary private key sk.

Great care must be taken when implementing this high level idea. First, clearly, the user cannot reconnect immediately because the server would then have a good guess who the real user it (in fact, if it's the only user connecting at that exact time, the server would know exactly who it is). Thus, some delay must be inserted. We therefore envisage a setting where the time is divided into slots and all users wishing to authenticate in any given slot are given each others' temporary public keys, and reconnect when the time slot concludes. This guarantees that anonymity is preserved with respect to all other users in the slot. Of course, this is not very practical unless many users continually connect so that a slot can be small (or unless the user can register its temporary public key well before it actually wants to work).

With some thought, it becomes apparent that the above solution actually intensifies the problem described above regarding whether the public keys or certificates of users that the server provides are actually real. We do not currently have any good solution for this.

5 Conclusions and Future Directions

There are a number of interesting open questions that arise out of what we have seen so far. First, more satisfactory solutions to anonymous authentication that can work with password-based technology are highly desirable. Second, the computational overhead of the anonymous authentication protocols that we have seen is such that both the user and server carry out ℓ public-key operations. Now, it is not difficult to see that the server must carry out this many public-key operations. However, it is not clear that the user must also do so.

Another open question relates to revocable anonymity. The protocols that we have seen assume that the user is given a smartcard that it has no control over. This automatically implies the use of dedicated hardware. In addition, the user has to trust the authenticating organization regarding its implementation. Thus, it is highly desirable to construct schemes that require only standard smartcard operations, and where the user anonymity can be guaranteed as long as the user runs correct software (that it can check or obtain from some open-source repository).

Acknowledgements

I would like to thank Tal Zarsky for everything he has taught me about privacy and for many helpful comments on this manuscript. I would also like to thank Vladimir Beker for helpful discussions regarding protocols for anonymous authentication.

References

[1] The Sunday Telegraph. Paedophiles calling a fifth of children on Net. June 2001. Can be found at http://www.telegraph.co.uk.

[2] Online Victimization: A Report on the Nation's Youth. By the Crimes Against Children Research Center (funded by the US Congress). The report can be downloaded from: http://www.missingkids.com/en_US/publications/NC62.pdf.

[3] Business Week/Harris Poll: A Growing Threat. The results of the poll can be found at http://www.businessweek.com/2000/00_12/b3673010.htm.

[4] M. Bellare, A. Boldyreva and S. Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In *Eurocrypt 2000*, Springer-Verlag (LNCS 1807), pages 259–274, 2000.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption. In *EUROCRYPT'94*, Springer-Verlag (LNCS 950), pages 92–111, 1994.

[6] D. Boneh and M.K. Franklin. Anonymous Authentication with Subset Queries. In 6*th ACM Conference on Computer and Communications Security*, pages 113–119, 1999.

[7] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT 2001*, Springer-Verlag (LNCS 2045), pages 93–118, 2001.

[8] D.L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[9] D.L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[10] D.L. Chaum and E. Van Heyst. Group Signatures. In *EUROCRYPT'91*, Springer-Verlag (LNCS 547), 1991.

[11] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO'98*, Springer-Verlag (LNCS 1462), pages 13–25, 1998.

[12] G. Danezis. The Traffic Analysis of Continuous-Time Mixes. *Privacy Enhancing Technologies Workshop*, 2004.

[13] D. Dolev, C. Dwork and M. Naor. Non-Malleable Cryptography. *SIAM Journal on Computing,* 30(2):391–437, 2000.

[14] D. Goldschlag, M. Reed and P. Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 42(2):39–41, 1999.

[15] D. Kesdogan, D. Agrawal and S. Penz. Limits of Anonymity in Open Environments. *5th Information Hiding Worskhop*, Springer-Verlag (LNCS 2578), 2003.

[16] J. Kilian and E. Petrank. Identity Escrow. In *CRYPTO 1998*, Springer-Verlag (LNCS 1462), pages 169–185, 1998.

[17] L.R. Knudsen and T. Kohno. Analysis of RMAC. In 10*th FSE*, Springer-Verlag (LNCS 2887), pages 182–191, 2003.

[18] M. Naor. Deniable Ring Authentication. In *CRYPTO 2002*, Springer-Verlag (LNCS 2442), pages 481–498, 2002.

[19] M.K. Reiter and A.D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[20] R.L. Rivest, A. Shamir and Y. Tauman. How to Leak a Secret. In *ASIACRYPT 2001*, Springer-Verlag (LNCS 2248), pages 552–565, 2001.

[21] S. E. Schechter, T. Parnell and A.J. Hartemink. Anonymous Authentication of Membership in Dynamic Groups. In 3*rd Financial Cryptography*, Springer-Verlag (LNCS 1648), pages 184–195, 1999.

[22] D.J. Solove. A Taxonomy of Privacy. *University of Pennsylvania Law Review*, Vol. 154, No. 3, January 2006.

[23] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[24] P. Syverson, G. Tsudik, M. Reed and C. Landwehr. Towards an Analysis of Onion Routing Security. *Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[25] S.D. Warren and L.D. Brandeis. The Right to Privacy. *Harvard Law Review*, Vol. IV, No. 5, December 15, 1890.

[26] T.Z. Zarsky. “Mine Your Own Business!”: Making the Case for the Implications of Data Mining of Personal Information in the Forum of Public Opinion. *Yale Journal of Law and Technology*, Vol. 5, 2002-2003.

[27] Y. Zhu, X. Fu, B. Graham, R. Bettati and W. Zhao. Anonymity Analysis of Mix Networks against Flow-Correlation Attacks, *Globecom Computer and Network Security*, 2005.

A Formal Definitions of Security

A.1 The Basic Definition

As we have mentioned, we will present more than one definition, for reasons that will become clear when we construct our protocols. The basic definition is a straightforward translation of the motivation presented in Section 4.1. As is standard for cryptographic protocols, we provide an asymptotic definition. That is, we consider probabilistic polynomial-time adversaries and we allow them to succeed with negligible probability. Recall that a function f is negligible if for every constant c there exists an integer n_c such that for every $n > n_c$ it holds that $f(n) < n^{-c}$. Thus, a negligible function is asymptotically smaller than every inverse polynomial. Intuitively, a negligible function is so small that we can just assume that events that occur with negligible probability never occur. In the definition below, we denote an anonymous authentication protocol Π to be a set of instructions for a user U, a server S, and a public-key generation algorithm G (G outputs pk and sk where pk is the public-key held by the server and sk is the private key held by the user). We therefore denote $\Pi = (G, S, U)$. We will require that Π be correct meaning that when all parties are honest, S accepts U and grants it access.

We begin by defining an experiment for anonymity between a corrupted server $\mathcal{A}$ and a user U_i from within a set of ℓ authorized users $U_1, \ldots, U_\ell$ (we denote the security parameter by n and for simplicity assume that this is the exact length of key):

The anonymity experiment $\mathsf{Expt}^{\mathsf{anon}}_{\Pi,\mathcal{A},\ell}(n)$:

1. *The key generation protocol G is run ℓ times. Denote the output to be $(pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)$ and let the i^{th} key be associated with U_i.*
2. *A random i is chosen uniformly from the set $\{1, \ldots, \ell\}$.*
3. *The adversarial server $\mathcal{A}$ is given $(pk_1, \ldots, pk_\ell)$ and interacts with U_i running Π and using private-key sk_i.*
4. *At the end of the experiment, $\mathcal{A}$ outputs an index $j \in \{1, \ldots, \ell\}$. The output of the experiment is defined to be 1, denoted $\mathsf{Expt}^{\mathsf{anon}}_{\mathcal{A},\mathcal{U},\Pi}(n) = 1$, if and only if $j = i$. In this case, $\mathcal{A}$ is said to have* succeeded.

Clearly, $\mathcal{A}$ can succeed with probability $1/\ell$ by just guessing j randomly. The definition below states that this is the best that $\mathcal{A}$ can do.

Definition 4 *A protocol $\Pi = (G, S, U)$ is said to achieve* perfect anonymity *if for every adversary $\mathcal{A}$ and every ℓ,*

$$\Pr\left[\mathsf{Expt}^{\mathsf{anon}}_{\Pi,\mathcal{A},\ell}(n) = 1\right] \leq \frac{1}{\ell} \ ,$$

We say that Π achieves computational anonymity *if for every probabilistic polynomial-time adversary $\mathcal{A}$ and every polynomial ℓ there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Expt}^{\mathsf{anon}}_{\Pi,\mathcal{A},\ell(n)}(n) = 1\right] \leq \frac{1}{\ell(n)} + \mathsf{negl}(n) \ .$$

Next, we proceed to define security against impersonation. We define this via an experiment involving a server S, a corrupted user $\mathcal{A}$, a parameter ℓ denoting the authorized users, and a security parameter n:

The impersonation experiment $\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n)$:

1. *The key generation protocol G is run ℓ times. Denote the output to be $(pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)$ and let the i^{th} key be associated with U_i.*
2. *The server S and adversary $\mathcal{A}$ are given $(pk_1, \ldots, pk_\ell)$, and then $\mathcal{A}$ interacts with S running Π and using public keys $pk_1, \ldots, pk_\ell$.*

3. *The output of the experiment is defined to be 1, denoted* $\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n) = 1$*, if at the end of the interaction with* $\mathcal{A}$ *the server* S *accepts (as instructed by* Π*). In this case,* $\mathcal{A}$ *is said to have* succeeded*.*

Unlike the case of anonymity, it is impossible to always prevent impersonation. For example, an adversary can always try to guess a private key. However, this should succeed with only negligible probability. We have the following definition:

Definition 5 *A protocol* $\Pi = (G, S, U)$ *is a* secure authentication protocol *if for every probabilistic polynomial-time adversary* $\mathcal{A}$ *and every polynomial* ℓ *there exists a negligible function* negl *such that*

$$\Pr\left[\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n) = 1\right] \leq \mathsf{negl}(n) \ .$$

We are now ready for our final definition:

Definition 6 *We say that* $\Pi = (G, S, U)$ *is a* secure protocol for anonymous authentication *if it is a correct and secure authentication protocol that achieves perfect or computational anonymity.*

A.2 A Weaker Definition – Verifiable Anonymity

The above definition captures the intuitive requirement of anonymity: no server should be able to know which user it is interacting with. However, in some cases we can relax this requirement. Specifically, consider the case of a cheating server who *can* learn the identity of the user. However, in the process of doing so, the user detects this cheating. In such a case, the user knows that her anonymity has been compromised and can just immediately logout. Since she hasn't done anything yet, no harm is done. In addition, the user now knows that this server cheats and so can reconsider working with it. We call this notion *verifiable anonymity* because the user can verify that her anonymity is preserved (that is, as long as she doesn't detect any attempt to cheat, she knows that her anonymity is guaranteed).

We now present this relaxed definition. The definition differs in that Π may now instruct a user to output a string cheat_S, indicating that the server S has cheated. We also modify the requirement of correctness so that when S and U are honest, it holds that S grants access to U, and U does *not* output cheat_S. We now define the verifiable anonymity experiment:

The verifiable anonymity experiment $\mathsf{Expt}^{\mathsf{verify}}_{\Pi,\mathcal{A},\ell}(n)$**:**

1. *The key generation protocol* G *is run* ℓ *times. Denote the output to be* $(pk_1, sk_1), \ldots, (pk_\ell, sk_\ell)$ *and let the* i^{th} *key be associated with* U_i*.*
2. *A random* i *is chosen uniformly from the set* $\{1, \ldots, \ell\}$*.*
3. *The adversarial server* $\mathcal{A}$ *is given* $(pk_1, \ldots, pk_\ell)$ *and interacts with* U_i *running* Π *and using private-key* sk_i*.*
4. *At the end of the experiment,* $\mathcal{A}$ *outputs an index* $j \in \{1, \ldots, \ell\}$*. The output of the experiment is defined to be 1, denoted* $\mathsf{Expt}^{\mathsf{anon}}_{\Pi,\mathcal{A},\ell}(n) = 1$*, if and only if* $j = i$ *and* U_i *does not output* cheat_S*. In this case,* $\mathcal{A}$ *is said to have* succeeded*.*

The definition of security remains the same. That is,

Definition 7 *We say that* $\Pi = (G, S, U)$ *is a* secure protocol for verifiable anonymous authentication *if it is a correct and secure authentication protocol that achieves perfect or computational verifiable anonymity.*

Note that in this definition the notion of "correctness" is different from the previous definition. We remark that correctness is needed in order to rule out the trivial protocols in which the user always outputs cheat_S or is never granted access.

B Formal Proof of Security of Protocol 1

Theorem 8 *Assuming that the public-key scheme used is secure again chosen-plaintext attacks, Protocol 1 is a secure protocol for verifiable anonymous authentication.*

Proof: The intuition behind the protocol is provided in Section 4.2.1 and we therefore proceed directly to the formal proof. First, it is clear that the protocol is *correct.* If both S and U_i are honest, then U_i will always return w and so will be granted access. In addition, S always encrypts the same w under all public keys and so U_i will never output cheat_S.

We now prove *perfect verifiable anonymity.* Let $\mathcal{A}$ be an adversarial server, let ℓ be a parameter, and let $c_1, \ldots, c_\ell$ be the ciphertexts sent by $\mathcal{A}$ to U_i in the first step of the protocol. Now, if U_i does not output cheat_S, then it must hold that there exist $r_1, \ldots, r_\ell$ such that for every $j = 1, \ldots, \ell$: $c_j = E_{pk_j}(w; r)$. By the correctness of encryption, this implies that for every $j = 1, \ldots, \ell$ we have $w = D_{sk_j}(c_j)$. Thus the view of $\mathcal{A}$ in the experiment is identical for any i chosen in experiment $\mathsf{Expt}^{\mathsf{verify}}_{\Pi,\mathcal{A},\ell}(n)$. This implies that the probability that $j = i$ *and* U_i does not output cheat_S is at most $1/\ell$, as required.

It remains to prove that Protocol 1 is a *secure authentication protocol.* We reduce this to the security of the encryption scheme against chosen-plaintext attacks. Let $\mathcal{A}$ be a probabilistic polynomial-time adversary and ℓ a polynomial, and let ϵ be a function such that

$$\Pr\left[\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n) = 1\right] = \epsilon(n) \ .$$

We use $\mathcal{A}$ to construct an adversary $\mathcal{A}_E$ that attacks the encryption scheme. We will show that $\mathcal{A}_E$ can succeed in the following experiment with probability $\epsilon(n)$: Adversary $\mathcal{A}_E$ is given ℓ public keys $pk_1, \ldots, pk_\ell$ and outputs two strings w_0 and w_1. A random bit b is then chosen and adversary $\mathcal{A}_E$ is given the ciphertexts $c_1 = E_{pk_1}(w_b), \ldots, c_\ell = E_{pk_\ell}(w_b)$. Following this $\mathcal{A}_E$ outputs a bit b', hoping that $b' = b$ (meaning that it guessed which message was encrypted). By the security of the encryption scheme, it holds that $\mathcal{A}_E$ can output $b' = b$ with probability that is at most negligibly greater than $1/2$ (this setting was formally studied in [4] although the specific experiment we consider is simpler and follows from the standard definition of security using a standard hybrid argument). We denote this experiment by $\mathsf{Expt}^{\mathsf{cpa}}_{\mathcal{A}_E}(n)$ and say that it equals 1 if and only if $\mathcal{A}_E$ output $b' = b$.

We now describe how $\mathcal{A}_E$ works: $\mathcal{A}_E$ receives keys $pk_1, \ldots, pk_\ell$ as input, and outputs two random strings w_0 and w_1. Adversary $\mathcal{A}_E$ then receives back ciphertexts $c_1, \ldots, c_\ell$. Given the above, $\mathcal{A}_E$ invokes $\mathcal{A}$ upon public keys $pk_1, \ldots, pk_\ell$ and simulates an execution of Π with $\mathcal{A}$ playing the user. In order to do this it just sends $\mathcal{A}$ the ciphertexts $c_1, \ldots, c_\ell$ as if they are the first message from the server to the user in Π. Then, $\mathcal{A}_E$ receives back a string w from $\mathcal{A}$. If $w \notin \{w_0, w_1\}$ then $\mathcal{A}_E$ outputs a random bit b'. Otherwise, if $w = w_0$ $\mathcal{A}_E$ outputs $b' = 0$ and if $w = w_1$ then $\mathcal{A}_E$ outputs $b' = 1$.

We now analyze the probability that $\mathcal{A}_E$ outputs $b' = b$. Setting Π to be Protocol 1, we have:

$$\Pr\left[\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n) = 1\right] = \Pr\left[\mathcal{A}(c_1, \ldots, c_\ell) = w\right]$$

where $c_1, \ldots, c_\ell$ are all encryptions of w (the above is rather informal because $\mathcal{A}$ is not invoked only on $c_1, \ldots, c_\ell$ but the intention is clear). For the sake of clarity, we denote $\Pr\left[\mathsf{Expt}^{\mathsf{cpa}}_{\mathcal{A}_E}(n) = 1\right] = \Pr[b' = b]$ (where b' and b are from the encryption experiment). Furthermore, when c_i is an encryption of w_0 we write c_i^0, and when it is an encryption of w_1 we write c_i^1. Thus $\mathcal{A}$ is given $(c_1^0, \ldots, c_\ell^0)$ when $b = 0$, and $(c_1^1, \ldots, c_\ell^1)$ when $b = 1$.

Now, consider the following three cases for any value of $b \in \{0, 1\}$:

1. If $\mathcal{A}$ outputs $w = w_b$ then $b' = b$ with probability 1;

2. If $\mathcal{A}$ outputs $w = w_{1-b}$ then $b' = b$ with probability 0;

3. If $\mathcal{A}$ outputs $w \notin \{w_0, w_1\}$ then $b' = b$ with probability 1/2;

Now, the probability that $\mathcal{A}$ outputs $w = w_b$ is exactly the probability that $\mathcal{A}$ succeeds in $\mathsf{Expt}^{\mathsf{imp}}_{\Pi,\mathcal{A},\ell}(n)$. Thus, this probability equals $\epsilon(n)$. Furthermore, the probability that $\mathcal{A}$ outputs $w = w_{1-b}$ is at most 2^{-n} because w_{1-b} is a uniformly distributed n-bit string that is never seen by $\mathcal{A}$ in the simulation by $\mathcal{A}_E$. Finally, the probability that $\mathcal{A}$ outputs $w \notin \{w_0, w_1\}$ is the complement of the first two events and so is at least $1 - \epsilon(n) - 2^{-n}$. We therefore have:

$$\begin{aligned}
&\Pr\left[b' = b\right] \\
&\quad = 1 \cdot \Pr\left[\mathcal{A}(c_1^b, \ldots, c_\ell^b) = w_b\right] + 0 \cdot \Pr\left[\mathcal{A}(c_1^b, \ldots, c_\ell^b) = w_b\right] + \frac{1}{2} \cdot \Pr\left[\mathcal{A}(c_1^b, \ldots, c_\ell^b) \notin \{w_0, w_1\}\right] \\
&\quad = \Pr\left[\mathcal{A}(c_1^b, \ldots, c_\ell^b) = w_b\right] + \frac{1}{2} \cdot \Pr\left[\mathcal{A}(c_1^b, \ldots, c_\ell^b) \notin \{w_0, w_1\}\right] \\
&\quad \geq \epsilon(n) + \frac{1}{2} \cdot (1 - \epsilon(n) - 2^{-n}) \\
&\quad = \frac{1}{2} + \frac{\epsilon(n)}{2} - \frac{1}{2^n}\ .
\end{aligned}$$

By the security of the encryption scheme in this experiment, we have $\epsilon(n)/2 - 2^{-n}$ must be negligible, which in turn implies that $\epsilon(n)$ is negligible. We conclude that $\mathcal{A}$ succeeds in the impersonation experiment with at most negligible probability, as required. This completes the proof. ■