A STUDY ON AUTHENTICATION MECHANISMS FOR OPEN DISTRIBUTED SYSTEMS

Dr. M.SURESH BABU
Professor, Department of CSE, Nalla Malla Reddy Engineering College, Hyderabad.

S.RAMCHANDRA REDDY
Asst. Professor, Department of CSE, Nalla Malla Reddy Engineering College, Hyderabad.

DEEPU JIMMY JOEL LAZARUS, Research Scholar, Department of CST, S.K.University, Anantapur, A.P.

ABSTRACT
While authentication within organizations is a well-understood problem, traditional Solutions are often inadequate at the scale of the Internet, where the lack of a central authority, the open nature of the systems, and issues such as privacy and anonymity create new challenges. For example, users typically establish dozens of web accounts with independently administered services under a single password, which increases the likelihood of exposure of their credentials; users wish to receive email from anyone who is not a spammer, but the openness of the email infrastructure makes it hard to authenticate legitimate senders; users may have a rightful expectation of privacy when viewing widely-accessed protected resources such as premium website content, yet they are commonly required to present identifying login credentials, which permits tracking of their access patterns. This paper describes enhanced authentication mechanisms to tackle the challenges of each of the above settings. Specifically, the paper develops: 1) a remote authentication architecture that lets users recover easily in case of password compromise; 2) a social network-based email system in which users can authenticate themselves as trusted senders without disclosing all their social contacts; and 3) a group access-control scheme where requests can be monitored while affording a degree of anonymity to the group member performing the request. The proposed constructions combine system designs and novel cryptographic techniques to address their respective security and privacy requirements both effectively and efficiently.

Keywords: *Authentication, Access Control, Cryptography*

1.0 INTRODUCTION
In its basic form, authentication is a well-understood concept in information security. Yet, many scenarios call for slight variations on the basic theme, where existing solutions do not directly apply; new techniques need to be developed. In the context of password-based user authentication, for example, users often reuse the same credentials (i.e., their passwords) when establishing accounts with dozens of independently administered services. Under such circumstances, a user whose password is compromised is unlikely to remember every place at which she needs to update her login information. At best, recovery from compromise is a lengthy, manual process. To attain its goals, the system employs proactive two-party signatures, a special kind of digital signatures, in which a private key is split between two parties, both of whom must approve and participate in signing authentication requests. This property enables a design in which an authentication server keeps a signature log describing all network accesses performed on behalf of the user, which provides a valuable audit trail in case of a break-in. Moreover, proactive two-party signatures allow private key shares to be updated, so that old shares cannot be combined with new ones to sign messages or to recover the private key.
While a number of proactive protocols have been proposed in the cryptography, they were all based on threshold schemes that cannot be applied to the practically relevant two-party case. Our novel construction fills this deficiency, providing a solution that is at the same time easy-to-implement and cryptographically secure.

1.1 Coping with Password Compromise in Web Authentication
This paper investigates proactive two-party signature schemes (P2SS) in the context of user authentication. P2SS allows two parties—the client and the server—jointly to produce signatures and periodically to refresh their sharing

of the secret key. The signature generation remains secure as long as both parties are not simultaneously compromised between successive refreshes. We construct the first such proactive scheme based on the discrete log assumption by efficiently transforming the popular Schnorr's signature scheme into a P2SS. We also extend our technique to the signature scheme of Guillou and Quisquater (GQ), providing two practical and efficient P2SSs that can be proven secure in the random oracle model under standard discrete log or RSA assumptions. We demonstrate the usefulness of P2SS (as well as our specific constructions) with a new user authentication mechanism for the Self-certifying File System (SFS). Based on a new P2SS signature protocol we call 2Schnorr, the new SFS authentication mechanism lets users register the same public key in many different administrative realms, yet still recover easily if their passwords are compromised. Moreover, an audit trail kept by a secure authentication server tells users exactly what file servers an attacker may have accessed—including even accounts the user may have forgotten about.

2.0 Methodology

In an ordinary two-party signature scheme, a private key is split between two parties, both of whom must approve and participate in the signing of messages. An attacker must compromise both parties to forge signatures on its own. However, the attacker has the entire lifetime of the public key to compromise each of the two parties. Moreover, particularly in the two-party case, the parties' roles may be asymmetric—for instance, a client may have the right to initiate signatures of arbitrary messages, while a server's role is simply to approve and log what has been signed. In such settings, an attacker may well gain fruitful advantage from the use of even a single key share, unless some separate mechanism is used for mutual authentication of the two parties. Finally, ordinary two-party signatures offer no way to transfer ownership of a key share from one party to another—as the old owner could neglect to erase the share it should no longer be storing.

Proactive digital signatures allow private key shares to be updated or "refreshed" in such a way that old key shares cannot be combined with new shares to sign messages or recover the private key. While a number of proactive signature protocols have been constructed, most existing protocols are threshold schemes designed for a variable number of parties. Because these threshold schemes require a majority of participants to be honest, they do not scale down to only two parties.

This paper describes 2Schnorr, a proactive signature protocol specifically designed for two parties. 2Schnorr is an efficient protocol that is easy to implement and produces digital signatures compatible with the Schnorr signature scheme. In the random-oracle model, a three-message version of 2Schnorr is provably secure against existential forgeries assuming only that discrete logs are hard. For applications with bounded concurrency, such as user authentication, a two-message version can also be proven secure under the stronger one-more-discrete-log assumption. The technique we describe can equally well be applied to the Guillou-Quisquater (GQ) signature scheme to produce two- and three-message 2GQ protocols based on the strong RSA and one-more-RSA inversion problems, respectively. To avoid redundancy in the treatment, though, this paper concentrates only on Schnorr signatures. Proactive two-party signature schemes (P2SS) have a natural application to the problem of user authentication, particularly in settings with many administrative realms. Within a large university, for example, it is not uncommon for a user to have five or six different shell accounts on machines in separate research groups. On the web, users typically establish accounts at dozens of different sites over time. Under such circumstances, a user whose private key or other credentials get compromised is unlikely to remember every place at which he needs to update his login information. Some of the sites may even be unavailable at the time the user tries to update them, at which point the user may just give up on the problem until the next time he

needs one of the Using 2Schnorr, we built a user-authentication mechanism that addresses these challenges for SFS. SFS is a secure, global file system in which users gain transparent access to files from many different administrative realms after logging in with a single password. With the new authentication mechanism, every user has an ordinary Schnorr public signature key on file wherever the user has an account. The corresponding private key is split between the user and an authentication server of the user's choice. If the user's password is ever compromised, he can immediately block further unauthorized access to all of his accounts by updating his password and private key halves on this single authentication server. Moreover, from the server's logs, the user can determine exactly what servers an attacker has accessed, where on the network those accesses came from, and whether the attacker has changed the user's login information at any sites. Thus, even accounts the user may have forgotten about will be brought back to his attention if there is any risk of an attacker having accessed them.

2.2 Related Work

A vast number of systems have dealt with the problem of user authentication. This section describes SFS and the motivation for a new SFS user authentication mechanism. We then highlight a few other systems that have tackled user authentication on a large scale.

2.2.1 SFS Overview

SFS is a secure network file system designed for decentralized control and easy sharing of files across organizational boundaries. In SFS's administrative model, servers are grouped into administrative realms that recognize the same set of authorized users. Realms can be as large as an entire campus or as small as a single server behind a DSL line. While a simple mechanism allows one realm to "import" or recognize users from another, realms in general need not trust each other, coordinate with each other, or even know of each other's existence. Each SFS user may have accounts in many different administrative realms. From a single client machine, users can simultaneously access servers in multiple realms. The SFS client itself has no notion of belonging to a particular realm. (In fact, SFS has no client-side configuration options that would differentiate one client from another.) Users simply access files based on whatever realms they belong to. If a user accesses a file on a server the client has never heard of, an "automounting" mechanism causes the file to spring into existence before the access completes.

SFS users have public signature keys which they register with any realms in which they have accounts. User authentication consists of digitally signing an authentication request with the corresponding private key. Each user runs a program, sfsagent, that attempts to authenticate her to every file server she accesses. In this way, by registering the same public key in every administrative realm, a user can transparently access files from multiple realms without worrying about administrative boundaries. Unfortunately, if a user's private key is ever compromised, the user may have to update her public key in a large number of realms. The mechanism described in this paper makes it considerably more difficult to compromise a user's key.

SFS comes bundled with a remote execution utility, rex, with similar functionality to the popular ssh. Between the file system and rex, any SFS user authentication mechanism can cover a large fraction of the day-to-day network accesses people make to their servers.

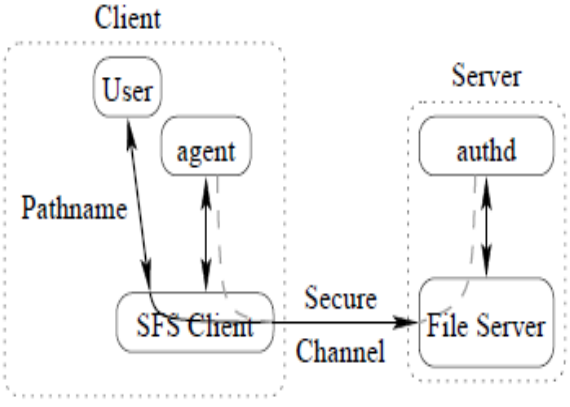

Fig1 :SFS User Authentication Architecture.

2.2.2 User Authentication

Of widely used network file systems, SFS's goals are probably most similar to those of AFS. AFS is a file system designed to work over the wide-area network. AFS has been particularly successful in large organizations—for instance permitting the user community of an entire university to share access to the same file systems. Unfortunately, AFS does not adapt as well to settings with many different administrative realms. AFS's security is based on the centralized Kerberos authentication system in which a central authority manages all of the accounts and servers in a given administrative realm. Cross-realm authentication is possible, but requires cooperation from realm administrators. Thus, users must typically type a separate password for each realm in which they wish to access servers. Since the central Kerberos server stores a secret that is effectively equivalent to the user's password, it is inadvisable for users to have the same password in different Kerberos realms.

The SSH remote login tool supports a mode of authentication based on public keys. The user registers his private key with an agent process on the local ma-chine, and stores the corresponding public key in a file .ssh/authorized_keys in his home directory on the server. SSH public key authentication is very convenient. Users therefore typically end up copying their authorized_keys file to all of their different accounts. Unfortunately, changing public keys requires many accounts to be updated, and users are likely to forget to update accounts on infrequently used machines.

Perhaps most relevant to P2SS are the various token- and hardware-based user-authentication systems. As smart cards and other physical security devices gain more computational power, it will become increasingly practical for them to compute digital signatures. Such configurations will be even more desirable if they can keep an audit trail of all signed messages in case the device is stolen or otherwise compromised. P2SS schemes enable such scenarios, while additionally allowing users to recover from compromised devices without changing their public keys. To compromise a user's public key permanently, an attacker would need to break the user's hardware device (or steal a backup of the user's share) and compromise the centralized signature server before the user had an opportunity to recover from the first event.

2.2.3 Two-Party Signature Generation

Ordinary two-party signature schemes are in some sense trivial. One can always take two copies of a secure one-party signature scheme (call them Sig_1 and Sig_2), publish the public keys pk_1, pk_2 for both of them, and let the first party store sk_1 and the second sk_2. A two-party signature of a message m then consists of two independent signatures of m using Sig_1 and Sig_2. The first signature can only be produced by the first party, and the second signature by the second party. Most previous work on two-party signatures has therefore focused on the problem of generating signatures that are compatible with existing one-party algorithms. Such two-party schemes allow systems to interoperate with verifiers that cannot be updated to understand new signature types. While 2Schnorr and 2GQ are the first two-party schemes compatible with Schnorr and GQ, they are hardly the first schemes to interoperate with standard one-party algorithms.

Bellare and Sandhu and MacKenzie and Reiter consider several flavors of two-party generation of the RSA (full domain hash) signature scheme (building on some previous less formal work, e.g.,). The schemes are simple, elegant, and in most cases reducible to the basic RSA assumption. MacKenzie and Reiter also give a protocol for two-party generation of DSA signatures. More closely related work was proposed in, where MacKenzie and Reiter extend their schemes from to allow for delegation of password-checking services. As noted by the authors, this extra property offers an approach to proactively update the password-protected secret key of a networked device. The resulting P2SS is designed and optimized for a hardware-based user authentication model, whereas the primary motivation of our study of P2SS is to obtain general-purpose schemes that could be combined with any user authentication mechanism.

2.2.4 Proactive Security

A basic two-party signature scheme remains secure only as long as both parties are not compromised. Unfortunately, the longer the lifetime of the public key, the more realistic the concern that both the client and the server may at one point have been compromised. General proactive cryptosystems address this problem by allowing potentially unbounded number of compromises, as long as not too many happen simultaneously. Specifically, there is an efficient share update protocol which allows players to refresh their current sharing of the secret key. As long as not too many servers are compromised between any two successive refreshes, the system remains secure. As with threshold cryptography, proactive cryptography has concentrated on $n \geq 3$ players. To the best of our knowledge, proactive signature schemes have not previously been studied in the two-party setting, except for the brief remark in the aforementioned work of. Recent work of Itkis and Reyzin on intrusion-resilient signatures describes a setting similar to P2SS. These combine the properties of key-insulated and proactive signatures. However, as in the key-insulated model, the server only helps the client to update its secret key from one time-period to another; all the signing is done by the client alone. In the P2SS model, the actual secret does not change from one time period to the next; only the sharing of the secret changes.

In both 2Schnorr and 2GQ, the share update protocol is very simple: a client simply sends a random element of an appropriate group to the server over a secure channel. Of course, this does not mean that proactivization is generally simple in the two-party case. Indeed, there seems to be no way to "proactivize" the generic double signature two-party approach, so the question of generic proactive two-party signature schemes is not as trivial as in the non-proactive case. Except for the two-party RSA scheme of, where proactivization comes at the cost of some efficiency loss (due to the need of resorting to the techniques of to share the secret over a much larger modulus), previous two-party signatures do not appear to "proactivize" in as simple way as our 2Schnorr and 2GQ schemes do.

2.3 The 2Schnorr Signing Protocol

This section specifies the 2Schnorr signing protocol and analyzes its security. Like the standard Schnorr signature scheme, 2Schnorr relies on a cryptographic hash function, H, which for the proofs we will assume behaves like a random oracle—an assumption very common in the cryptographic research, first formalized in. Before going into the details of 2Schnorr, we briefly describe the standard Schnorr scheme.

The Schnorr signature scheme was first proposed in as an application of the Fiat-Shamir transformation, and its security has been analyzed, among the others, in. It can be instantiated on every group G of prime order where the discrete log problem is believed to be hard, and can be proven secure under the standard notion of existential unforgeability against the adaptive chosen-message attack in the Random Oracle Model, assuming that computing discrete logs in the underlying group is hard. For concreteness, we will consider cyclic subgroups of Z^*_p (for large primes p) of prime order q.

The key generation algorithm produces two large prime p and q such that $q|(p-1)$, and an element g of Z^*_p of order q. Then it picks a random element x in Z^*_q, and sets $y = g^x \bmod p$. The public key is now (p, q, g, y), while the corresponding private key is x. Notice that the group parameters p, q, g can be safely shared between a community of users, so that y by itself can be thought as the public key corresponding to the private key x. We will also assume that a cryptographic hash function H mapping arbitrary strings into elements of Z^*_q has been specified as a parameter of the scheme.

To sign a message m, the holder of the private key x picks a random $k \in Z^*_q$ and set $r = g^k \bmod p$. It then computes $e = H(m, r)$, $s = k + xe \bmod q$, and outputs the signature (r, s). Notice that k must be kept secret and chosen anew each time: disclosing or reusing the value of k would allow recovery of the secret key x.

To check whether a given (r, s) is indeed a signature for some message m, it suffices to know the corresponding public key (p, q, g, y) and verify that $g^s = ry^e \bmod p$, where e = H(m, r).
The Schnorr signature scheme is often studied together with the GQ scheme, its "twin" based on the RSA assumption. In fact, they can be thought as variations of the same basic theme. Semantically, the difference between the two schemes is that in Schnorr's all secrets are drawn from the additive group $(Z_q,+)$ and their public counterparts are obtained by exponentiation on a fixed base; in the GQ scheme, instead, private data is taken from the multiplicative group $(Z_{*N}, *)$ (where N is the product of two big primes) and public quantities are obtained by exponentiating to a fixed exponent. Syntactically, this is equivalent to convert all additions in Schnorr's scheme into multiplications, and all multiplications (involving secrets) into exponentiations. A mechanical application of such "conversion rules" to our 2Schnorr protocol (described below) yields the 2GQ protocol, which enjoys analogous security properties based on the RSA assumption.

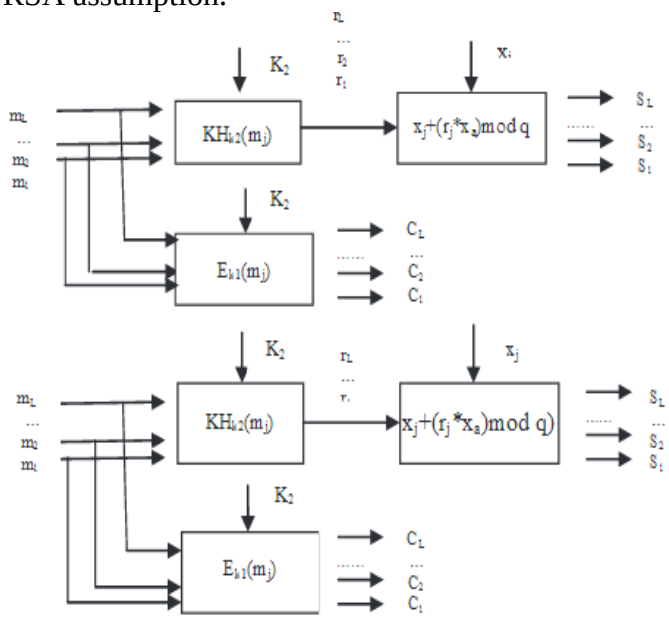

Fig 2 : Parallelization of Schnorr Signcryption algorithm.

The main issue in obtaining a two-party solution for Schnorr signatures is that if one party (say the client) could control the choice of one of the secret quantities x or k, or their public counterparts y and r, then the client would gain an advantage over the server, and the resulting scheme might not be secure. Fortunately, in our case the parties need just to agree on a random value, a task usually referred to as coin flipping. This can easily be achieved by having each party choosing its random share, and then exchanging and combining the two shares. To avoid any unfairness in the exchange due to the fact that one party (say the server) must reveal its share first, the server will only "commit" to its random share, and will "open" the commitment only upon receiving the client's random value.

3.0 E-Mail Authentication via Social Networks

A variety of peer-to-peer systems use social networks to establish trust between participants. Yet the sharing of social information introduces privacy concerns. This paper describes new privacy-preserving cryptographic protocols that enable participants to verify social proximity while exposing minimal information about the parties' social contacts. Compared to previous results, our protocols are either significantly more efficient (orders of magnitude faster than the private-matching approached used in PM) or achieve stronger security properties at similar cost.

3.1 Privacy Preserving Cryptographic Protocols

In peer-to-peer systems where resources are scarce or users are subject to abuse, participants can leverage social relationships to guide their interactions with other users. Further considering transitive trust relationships can extend a user's vantage, while still incurring a low risk of coming across abusive users. In the email or instant messaging contexts, for example, social networks can facilitate cooperative spam blacklisting or sender whitelisting . A naive approach to discover transitivity is for one party to send his list of friends to the other party, who computes the set intersection of their two input sets. Yet this simple form of information sharing introduces privacy concerns. While the problem of privacy-

preserving two-party computation has been widely studied in the cryptographic literature, general-purpose cryptographic solutions are too computationally expensive for practical use. Furthermore, their privacy guarantees are often misaligned with applications' specific threat models

This paper describes efficient cryptographic protocols with which parties can determine shared friends while exposing minimal information about their social contacts. Using Re: as a motivating example—an email system that reliably accepts mail from senders based on proximity in a social network—we describe two alternative methods to verify social proximity. The first method, based only on cryptographic hash functions and symmetric encryption, meets all of Re:'s current privacy and security goals at a fraction of the cost of its current Private Matching protocol. The second method, while of comparable cost, achieves stronger privacy guarantees (namely, non-transferability) through its novel use of cryptographic properties of bilinear groups.

Our contributions are twofold. First, we describe and define a security model for verifying social connectedness in a privacy-preserving fashion. In fact, the mismatch between Re:'s goals and the privacy properties offered by Private Matching were a source of both computational inefficiency and privacy limitations. Second, we propose cryptographic protocols that protect such social

3.2 Motivating application: Re:

Reliable Email (Re:) is an automated email acceptance system that whitelists email according to its sender. It seeks to undue the email unreliability introduced by content-based filters and other spam-fighting technologies which, while seeking to minimize the amount of spam that reaches a user's inbox, occasionally misclassify legitimate mail as spam.

The concept of sender-based whitelisting for email is hardly new. Yet, traditional whitelists suffer from two chief usability issues. First, a recipient's whitelist cannot accept mail from a sender previously unknown to the recipient. Second, populating whitelists requires manual effort distributed diffusely in time, as users acquire new contacts. To overcome these limitations, Re: automatically broadens the set of senders whose mail is accepted by recipients' whitelists by explicitly examining the social network among email users. Specifically, Re: allows a user R to attest to another user S, which indicates that R is willing to have email from S directly forwarded to his mailbox. In other words, "User R trusts his friend S not to send him spam." Such an attestation is a digitally-signed statement of the form:[1] $\sigma_{R \to S} = \{H(R), H(S), \text{start}, \text{end}\}_{SK_R}$ where H is a collision-resistant cryptographic hash function like SHA-256 operating on the users' email addresses, start and end define the attestation's

4.0 Authentication and Privacy for Group Access Control

This paper introduces Ad Hoc Anonymous Identification schemes, a new multiuser cryptographic primitive that allows participants from a user population to form ad hoc groups, and then prove membership anonymously in such groups. Our schemes are based on the notion of accumulator with one-way domain, a natural extension of cryptographic accumulators we introduce in this work. We provide a formal model for Ad Hoc Anonymous Identification schemes and design secure such schemes both generically (based on any accumulator with one-way domain) and for a specific efficient implementation of such an accumulator based on the Strong RSA Assumption. A salient feature of our approach is that identification protocols take time independent of the size of the ad hoc group. All our schemes and notions can be generally and efficiently amended so that they allow the recovery of the signer's identity by an authority, if the latter is desired.

Via the Fiat-Shamir transform, we obtain constant-size, signer-ambiguous group and ring signatures (provably secure in the Random Oracle Model). For ring signatures, this is the first such constant-size scheme, as all the previous proposals had signature size proportional to the size of the ring. For group signatures, we obtain

schemes comparable in performance with state-of-the-art schemes, with the additional feature that the role of the group manager during key registration is extremely simple and essentially passive: all it does is accept the public key of the new member (and update the constant-size public key of the group).

4.1 Multiuser Cryptographic Primitive

Anonymous identification is an oxymoron with many useful applications. Consider the setting, for a known user population and a known set of resources, where a user wants to gain access to a certain resource. In many cases, accessing the resource is an action that does not mandate positive identification of the user. Instead, it would be sufficient for the user to prove that he belongs to the subset of the population that is supposed to have access to the resource. This would allow the user to lawfully access the resource while protect his real identity and thus "anonymously identify" himself. Given the close relationships between identification schemes and digital signatures, one can easily extend the above reasoning to settings where a user produces a signature that is "signer-ambiguous" i.e., such that the verifier is not capable of distinguishing the actual signer among a subgroup of potential signers. In fact, it was in the digital signature setting that such an anonymous scheme was presented for the first time, with the introduction of the group signature model, which additionally mandates the presence of a designated party able to reveal the identity of the signer, were the need to arise.

Subsequent work on group signatures and on anonymous identification in general allowed for more efficient designs and formal modelling of the primitive, with the current state of the art being the scheme by Ateniese et al.. In general, existing group signature schemes are derived from their interactive counterpart (ID Escrow schemes) via the Fiat-Shamir transform. A related notion, but of slightly different nature, is that of ring signatures, introduced by Rivest, Shamir and Tauman in and further studied in. Ring signatures differ from group signatures in that they allow group formation to happen in an ad hoc fashion: group must be formed without the help of a group manager; in fact, a user might not even know that he has been included in a certain group. This is in sharp contrast to the group signature setting where the user must execute a Join protocol with the group manager and obtain a group-membership certificate that cannot be constructed without the help of the group manager. Note that ad hoc group formation in the context of ring signatures is always understood within the context of a user population and an associated PKI. Based on the PKI, ad hoc subsets of the user population can be formed without the help of a "subset manager"—but it is assumed that every user has a registered public key.

While ring signatures are attractive because they have simple group formation procedures that can be executed by any user individually, they have the shortcoming that the length of the signature is proportional to the group size. For large groups, the length of a ring signature (growing linearly with the group size) will become impractical. To the contrary, schemes with constant- size signatures have been successfully designed in the group signature setting.

We remark that in the setting of anonymous identification, the counterpart of "signature size" is the bandwidth consumed by the protocol, which is thus an important complexity measure to minimize. Based on the above discussion, an important open question in the context of anonymous identification and signature schemes, recently posed by Naor in, is the following: Is it possible to design secure anonymous identification schemes that enable ad hoc group formation in the sense of ring signatures and at the same time possess constant-size signature (or proof) length? This paper answers the above question in the affirmative. Specifically, we introduce a new primitive called Ad Hoc Anonymous Identification schemes; this is a family of schemes where participants from a user population can form groups in ad hoc fashion (without the help of a group manager) and then get anonymously identified as members of such

groups. Our main tool in the construction of Ad Hoc Anonymous Identification schemes is a new cryptographic primitive, accumulator with one-way domain, which extends the notion of a collision-resistant accumulator. In simple terms, in an accumulator with one-way domain, the set of values that can be accumulated are associated with a "witness space" such that it is computationally intractable to find witnesses for random values in the accumulator's domain.

First, we demonstrate the relationship between such accumulators and Ad Hoc Anonymous Identification schemes by presenting a generic construction based on any accumulator with one-way domain. Second, we design an efficient implementation of accumulator with a one-way domain based on the Strong RSA Assumption, from which we obtain a more efficient construction of Ad Hoc Anonymous Identification scheme whose security rests upon the Strong RSA Assumption. We remark that previous work on anonymous identification that allowed subset queries was done by Boneh and Franklin. They define a more limited security model, and show a protocol which imposes on both parties a computational load proportional to the subset size at each run. Moreover, their scheme is susceptible to collusion attacks (both against the soundness and against the anonymity of the scheme) that do not apply to our setting schemes. We present a general construction for our primitives from any accumulator and not just the one of. Our formal definitional framework is of independent interest.

Conclusion

As stated before the normal Internet user has too many username and password combinations to remember which leads to insecure passwords. In our Strong-RSA-based Ad Hoc Anonymous Identification scheme, the computational and communication complexity on both ends is constant, regardless of the size of the group. Thus, the signature version of our ad hoc anonymous identification scheme yields a ring signature with constant size signatures (over a dedicated PKI). Other applications of our scheme include "ad hoc group signatures" (group signature schemes where the group manager can be offline during the group formation) and identity escrow over ad hoc groups. Building on work by Camenisch and Lysyanskaya, Tsudik and Xu investigated techniques to obtain more flexible dynamic accumulators, on which to base a group signature scheme (which is one of our applications). The specific method used by bears many similarities with our Strong-RSA-based instantiation, with some important differences. Namely, in their solution anonymity revocation takes time proportional to the user population, due to subtle problems concerning the accumulation of composite values inside the accumulator. Our work resolves this technical problem. Moreover, we present a new notion of Ad Hoc Anonymous Identification scheme, which has more applications than those specific to group signature schemes

Bibliography

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Advances in Cryptology—Crypto'00, volume 1880 of Lecture Notes in Computer Science, pages 255–270, Berlin, 2000. Springer.

[2] G. Ateniese and B. de Medeiros. Efficient group signatures without trapdoors. In Advances in Cryptology—ASIACRYPT '03, volume 2894 of LNCS, pages 246–268. Springer, 2002.

[3] M. Bellare, M. Fischlin, S. Goldwasser, and S. Micali. Identification protocols secure against reset attacks. In Advances in Cryptology—EUROCRYPT '01, volume 2045 of LNCS, pages 495–511. Springer, 2001.

[4] J. Benaloh and M. de Mare. One-way accumulators: a decentralized alternative to digital signatures. In Advances in Cryptology—EUROCRYPT '93, volume 765 of LNCS, pages 274–285. Springer, 1993.

[5] D. Bleichenbacher. Generating ElGamal signatures without knowing the secret key. In Advances in Cryptology—EuroCrypt'96, volume 1070 of Lecture Notes in Computer Science, pages 10–18, Berlin, 1996. Springer-Verlag.

[6] M. Blum. Coin flipping by telephone. In IEEE Spring COMPCOM, pages 133–137, 1982.
[7] A. Boldyreva. Efficient threshold signature, multisignature and blind signature schemes based on the gap-diffie-hellman-group signature scheme. In Public Key Cryptography—PKC'03, volume 2567 of Lecture Notes in Computer Science, pages 31–46, Berlin, 2003. Springer-Verlag. Full version available at http://eprint.iacr.org/2002/118/.
[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Advances in Cryptology—EUROCRYPT '03, volume 2656 of LNCS, pages 416–432. Springer, 2003.
[9] D. Chaum and E. van Heyst. Group signatures. In Advances in Cryptology—EUROCRYPT '91, volume 547 of LNCS, pages 257–265. Springer, 1991.
[10] L. Chen and T. P. Pedersen. New group signature schemes (extended abstract). In Advances in Cryptology—EUROCRYPT 94, volume 950 of LNCS, pages 171–181. Springer, 1994.
[11] FIPS 180-1. Secure Hash Standard. U.S. Department of Commerce/ N.I.S.T., Springfield, VA, April 1995.
[12] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, 18(1):186–208, 1989.
[13] J. Kilian and E. Petrank. Identity escrow. In Advances in Cryptology—CRYPTO '98, volume 1462 of LNCS, pages 169–185. Springer, 1998.
[14] J. Kong, P. O. Boykin, B. Rezaei, N. Sarshar, and V. Roychowdhury. Let your cyberalter ego share information and manage spam, May 2005. http://arxiv.org/abs/physics/0504026.
[15] C. Schnorr. Efficient signature generation by smart cards. Journal of Cryptology, 4(3):161–174, 1991.
[16] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In Proceedings of the Winter 1988 USENIX, pages 191–202, Dallas, TX, 1988. USENIX.