Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices

R. Gennaro*, D. Leigh**, R. Sundaram***, and W. Yerazunis†

Abstract. We present a *batch* version of Schnorr's identification scheme. Our scheme uses higher degree polynomials that enable the execution of several Schnorr's protocol at a cost very close to that of a single execution. We present a full proof of security that our scheme is secure against impersonation attacks.

The main application of this result is a very efficient way for a party to prove that it holds several secret keys (i.e. identities), where each identity is linked to a specific authorization. This approach protects the privacy of the prover allowing her to prove only the required set of authorizations required to perform a given task, without disclosing whether she is in possession of other privileges or not.

We also show that our scheme is suitable to be implemented on low-bandwidth communication devices. We present an implementation of a smart card employing recent technology for the use of LEDs (Light Emitting Diodes) for bidirectional communication. Another contribution of our paper is to show that this new technology allows the implementation of strong cryptography.

1 Introduction

Identification, also known as entity authentication, is a process by which a *verifier* gains assurance that the identity of a *prover* is as claimed, i.e. there is no impersonation [MOV97,Sch96]. An identification scheme enables a prover holding a secret key to identify itself to a verifier holding the corresponding public key.

The primary objectives of an identification protocol are *completeness* - in the case of honest parties the prover is successfully able to authenticate itself to the verifier, and *soundness* - a dishonest prover has a negligible probability of convincing a verifier.

* IBM T.J.Watson Research Center, Yorktown Heights, NY, USA. rosario@watson.ibm.com

** Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. leigh@merl.com

*** Northeastern University, Boston, MA, USA. This research funded by MERL. koods@ccs.neu.edu

† Mitsubishi Electric Research Laboratories, Cambridge, MA, USA. yerazunis@merl.com

There are various grades of dishonesty and corresponding levels of security. The goal of the adversary is to impersonate the prover. As per the standard security framework [FFS88], the adversary is allowed various attacks on the honest prover, which complete before the impersonation attempt. A typical requirement of identification protocols is that they be secure against impersonation under *passive* attack, where the adversarial prover has access to transcripts of prover-verifier interactions. A stronger requirement is that protocols be secure against *active* attacks where the adversarial prover can actively play the role of a cheating verifier with the prover numerous times before the impersonation attempt. Security against impersonation under active attack has been the traditional goal of identification schemes.

However, in recent times interest has been growing in still stronger attacks, e.g. *concurrent attacks*. In these attacks, just like with active attacks the adversarial prover gets to play the role of cheating verifier prior to impersonation with the key distinction that the adversary is allowed to interact with multiple honest prover clones concurrently [FFS88].

It is very important to keep in mind that, in the real world, identification protocols provide assurances only at the instant of time when the protocol is successfully completed. It is therefore important to ensure that the identification process is tied to some form of ongoing *real-world* integrity service. At some level all identification schemes are vulnerable to the adversary who cuts in immediately after the successful identification of the legitimate party.

Zero-knowledge Protocols. A paradigm introduced in [FFS88] to construct identification protocols, is to construct *zero-knowledge proofs of knowledge*. These are protocols which allow a prover to demonstrate knowledge of a secret while revealing no information whatsoever other than the one bit of information regarding the possession of the secret [GMR85,FFS88].

A protocol is said to be *honest-verifier zero-knowledge* if it is zero-knowledge when interacting with honest verifiers. An honest-verifier zero-knowledge protocol has a weaker security guarantee than a general zero-knowledge protocol since it is possible that a dishonest verifier can extract information from the prover in the former protocol.

However, when used as identification schemes, the ultimate measure of the worth of a protocol lies in its security against impersonation attempts and a protocol that is secure against impersonation against concurrent attacks is considered to be "secure" even if it is “only” honest-verifier zero-knowledge. This happens if one is able to show that whatever information is leaked to the dishonest verifier, it does not help him in any impersonation attack.

Schnorr in [Sch91] presents such a protocol, based on the hardness of computing discrete logarithms. The details are described in the body of the paper, but here we just remark that Schnorr’s protocol is an honest-verifier zero-knowledge proof of knowledge of a discrete logarithm. Recently Bellare and Palacio in [BPa02] showed that under a slightly stronger assumption on the security of discrete logarithms, Schnorr’s protocol is a secure identification scheme against concurrent attacks.

1.1 Authorization

Authorization is the conveyance to the verifier that the prover, has the sanction to gain access to a particular resource set, or belongs to a certain privilege class. Authorization may be effected by the proving of one or of multiple identities.

Consider now the following access control scenario. Users of a given system belong to various privilege classes. Access control classes for the data are defined using these privileges, i.e. as the users who own a given subset of privileges. For example the access control class for a given piece of data D, is defined as the users who own privileges P_1, P_2, P_3.

A way to implement such an access control system is to give each user a certified public key. The certificate would indicate the subset of privileges associated with this public key. Then in order to gain access, Alice performs an identification protocol based on her public key, and if her privileges are a superset of the ones required for the access she is attempting, access is granted.

There are several drawbacks with this approach. But the main one is a blatant violation of Alice's privacy. Whenever Alice proves her identity she reveals *all* her privileges, when, theoretically, in order to gain access she should have had to reveal only a subset of them.

It is clear that there are situation which warrant a privacy-preserving authorization mechanism, in which Alice can gain access by proving she owns the minimal set of required privileges.

This can be done by associating a different public key to each privilege. Then Alice would prove that she knows the secret keys required for the authorization. Using typical proofs of knowledge, like Schnorr's, to prove knowledge of k keys the user has to perform k proofs. Although these proofs can be performed in parallel, keeping the round complexity the same, the computational complexity goes up by a factor of k.

Thus an interesting question is if it is possible to perform a proof of knowledge of d secrets at the cost of less than d proofs. We answer this question in the affirmative (see below).

Another advantage of associating different keys to different privileges, is that the latter can be easily transferred simply by transferring the corresponding secret key.

1.2 Our Contributions

We present a *batch* version of Schnorr's protocol. In our scheme the prover can prove knowledge of d secret keys (discrete logarithms), at a cost slightly superior to the cost of a single Schnorr's protocol, thus saving a factor of d in computation and bandwidth over the best previously known solutions. We use degree d polynomials to represent an ordered list of d identities. We show that the resulting scheme is not only honest-verifier zero-knowledge, but that it is also a secure identification scheme against impersonation under concurrent attacks [BPa02].

This immediately yields a very efficient privacy-preserving authorization mechanism along the lines described in the previous section.

Finally, in order to showcase the efficiency of our proposal we present an implementation in a very low-bandwidth environment. We use recently proposed technology to use Light Emitting Diodes (LEDs) as bi-directional communication devices. We believe that another interesting contribution of our paper is to show that this new technology is robust enough to implement strong cryptographic solutions.

1.3 Related work

Besides the works cited above [GMR85,FFS88,Sch91], another widely used protocol for identification is the one proposed by Guillou and Quisquateur in [GQu88]. This scheme is more efficient than Schnorr's and very suitable to low-power computation devices. When proving multiple identities simultaneously, our batch technique makes the advantage of using GQ over Schnorr's disappear very quickly. Indeed only for very small values of d (the number of identities being proven), d parallel executions of GQ beat our batch Schnorr protocol in efficiency[1]. It would be interesting to devise a batch version of the Guillou-Quisquateur protocol, but we were not able to do so.

In any case, when comparing our scheme with running d Guillou-Quisquateur schemes, one should remember that the GQ scheme is based on a different assumption (RSA inversion) than the Schnorr's protocol.

Our new identification scheme is related to the concept of batch verification of signatures [BGR98]. As far as we know there has not been any work on batch verification for identification protocols. A straightforward application of the techniques in [BGR98] to our problem would yield a much less efficient protocol. Moreover, the mathematical techniques we use are fundamentally different than the ones in [BGR98].

Recently the area of privacy-preserving protocol has received a lot of attention. We refer the reader especially to the works by Camenisch and Lysyanskaya [CL01,CL02], where the concept of *group signature* is used to show how a user can prove membership in a certain privilege class, without revealing her true identity. These solutions offer a very strong privacy guarantee, as a user can safely prove his privileges to various verifiers, who would not be able to *link* her various transactions. On the other hand our solution does not protect the identity of the user, but simply allows her to prove the minimal set of privileges required for a given transaction. But if verifiers collude they can link the user's transactions and reconstruct the set of privileges she holds. For example if Alice proves to Bob that she belongs to privilege class P1, and to Charles that she

[1] Jumping ahead, assume we perform Schnorr's scheme, with parameters p, q such that $|p| = 1024$ and $|q| = 160$. Then one execution of the protocol costs about 240 multiplications for the prover (i.e. one exponentiation mod$p$, with a 160-bit exponent). On the other hand, if we perform the GQ scheme over a 1024-bit RSA modulus, using a small public exponent (like 3), and security parameter 80, then the prover's cost is about 80 multiplications. Thus GQ is approximately 3 times as fast as Schnorr's. Which means that for $d > 3$, i.e. when proving more than 3 identities simultaneously, our batch Schnorr protocol becomes more attractive than GQ.

belongs to the P2 class, it is possible for Bob and Charles together to understand that Alice holds both P1 and P2 privileges. On the other hand our solution is much simpler and more efficient that solutions based on group signatures, thus it could be preferable in a scenario in which collusion is not really a problem. Moreover some of our batching techniques can be used to speed-up solutions based on group signatures (since there, the proof of possession of several keys is a subprotocol).

2 Preliminaries

In this section we recall the basic definition of proof of knowledge, the computational assumptions that we are going to need, and Schnorr's protocol. In the following the acronym PPT stands for "probabilistic polynomial-time".

2.1 Proofs of Knowledge

Polynomial Time Relationships. Let $\mathcal{R}$ be a polynomial time computable relationship, i.e. a language of pairs (y, w) such that it can be decided in polynomial time in $|y|$ if $(y, w) \in \mathcal{R}$ or not. With $\mathcal{L}_\mathcal{R}$ we denote the language induced by $\mathcal{R}$ i.e. $\mathcal{L}_\mathcal{R} = \{y \,:\, \exists w \,:\, (y, w) \in \mathcal{R}\}$.

More formally an ensemble of polynomial time relationships $\mathcal{PTR}$ consists of a collection of families $\mathcal{PTR} = \cup_n \mathcal{PTR}_n$ where each $\mathcal{PTR}_n$ is a family of polynomial time relationships $\mathcal{R}_n$. To an ensemble $\mathcal{PTR}$ we associate a randomized *instance generator* algorithm IG that on input 1^n outputs the description of a relationship $\mathcal{R}_n$. In the following we will drop the suffix n when obvious from the context.

Example: The instance generator algorithm on input 1^n outputs an n-bit prime q, a *poly*(n)-prime p, such that $q|p-1$ and an element g of order q in Z_p^*. The corresponding relationship is that of pairs $(y, w) \subset Z_p^* \times Z_q$ such that $y = g^w \bmod p$.

Proofs of Knowledge. In a proof of knowledge for a relationship $\mathcal{R}$, two parties, Prover P and Verifier V, interact on a common input y. The Prover also holds a secret input w, such that $(y, w) \in \mathcal{R}$. The goal of the protocol is to convince V that P indeed knows such w. Ideally this proof should not reveal any information about w to the verifier, i.e. be zero-knowledge.

The protocol should thus satisfy certain constraints. In particular it must be *complete*: if P knows w then V should accept. It should be *sound*: for any (possibly dishonest) prover who does not know w, the verifier should almost always reject. Finally it should be *zero-knowledge*: no (poly-time) verifier (no matter what possibly dishonest strategy she follows during the proof) can learn any information about w.

A formal definition of proofs of knowledge can be found in [BGo93] (improving on the original definition in [FFS88]). Informally, the concept of "knowing" w is formalized by showing that w can be computed in polynomial-time if we have black-box access to P. This is done by constructing a *witness extractor* which runs in probabilistic polynomial time, and computes w with a probability related to the probability that the Prover makes the Verifier accept.

The concept of zero-knowledge is formalized via the existence of a probabilistic polynomial time simulator S that on input y and interacting with a possibly cheating Verifier outputs transcripts with the same probability distribution as the real prover (who knows w).

A formal definition follows. With $[\mathsf{P}(y,w),\mathsf{V}(y)]$ we denote the output of the protocol, i.e. 1 iff V accepts. With $\pi_{\mathsf{P}}(n)$ we denote the probability that a prover P makes the verifier accept, i.e.

$$\pi_{\mathsf{P}}(n) = Prob[\ \mathcal{R}_n \leftarrow \mathsf{IG}(1^n)\ ;\ [\mathsf{P}(y,\cdot),\mathsf{V}(y)] = 1\ ]$$

where the statement y can be chosen by P.

Definition 1. *We say that* (P,V) *is a proof of knowledge for a relationship* $(\mathcal{PTR},\mathsf{IG})$ *if the following properties are satisfied:*

Completeness *For all* $(y,w) \in \mathcal{R}_n$ *(for all* $\mathcal{R}_n$*) we have that* $[\mathsf{P}(y,w),\mathsf{V}(y)] = 1$*.*

Witness Extraction *There exist a probabilistic polynomial time* knowledge extractor KE*, a function* $\kappa : \{0,1\}^* \rightarrow [0,1]$ *and a negligible function* ϵ*, such that for all PPT* P'*, if* $\pi_{\mathsf{P}'}(n) > \kappa(n)$ *then* KE*, given rewind access to* P'*, computes* w *such that* $(y,w) \in \mathcal{R}_n$ *with probability at least* $\pi_{\mathsf{P}'}(n) - \kappa(n) - \epsilon(n)$*.*

Zero-Knowledge *For every PPT Verifier* V' *there exist a probabilistic polynomial time* simulator $\mathsf{SIM}_{\mathsf{V}'}$*, such that for all* $(y,w) \in \mathcal{R}_n$ *the two random variables*

$$\mathsf{View}[\mathsf{P}(y,w),\mathsf{V}'(y)]$$
$$\mathsf{View}[\mathsf{SIM}_{\mathsf{V}'}(y),\mathsf{V}'(y)]$$

are indistinguishable.

The function κ is called the *knowledge error* and measures the probability, inherent to the protocol, that a cheating prover can convince the verifier without knowing w. What we require is that if a prover convinces the verifier with a probability higher than κ, then we can extract the witness with a success probability related to the difference.

2.2 Identification Schemes

In an identification scheme a prover P and a verifier V interact on input a public key (generated together with its matching secret key by a key generation algorithm KG). The prover holds the matching secret key, and his goal is to convince the Verifier of this fact, and thus of his identity.

An *impersonation attack* is when an adversary $\mathcal{A}$ tries to convince the verifier that he is the honest prover. This kind of attack is called a *passive attack*, if the adversary attempts impersonation only after having witnessed several correct executions of the identification protocol between the prover and honest verifiers. We said that an attack, is *active*, if the adversary before trying to impersonate the prover has engaged with him in the identification protocol, playing the role of a (possibly dishonest) verifier.

Finally, and this is the notion we consider in this paper, we say that an active impersonation attack is a *concurrent attack* if the interactions between the adversary and the honest prover before the impersonation attack, can be carried out in a concurrent fashion (i.e. with an arbitrary scheduling of messages).

So we can consider the following game. A pair of keys sk,pk is chosen according to the distribution induced by KG on input 1^n the security parameter. The prover is given sk and pk is made public. Then the adversary $\mathcal{A}$ engages as a verifier in several concurrent executions of the identification protocol with the prover. He then finally runs one execution of the protocol, as the prover, with an honest verifier. We denote with $\mathsf{adv}_{\mathcal{A}}(n)$ the probability that $\mathcal{A}$ makes the verifier accept at the end of this game.

Definition 2. *We say that an identification protocol is secure against concurrent impersonation attack if* $\mathsf{adv}_{\mathcal{A}}(n)$ *is negligible in* n*.*

2.3 Discrete Logarithm Assumptions

Since its introduction in the seminal paper by Diffie and Hellman [DH78], the discrete logarithm assumption has been widely used to construct cryptographic algorithms. Here we are going to use a well established variant of the assumption that considers the hardness of computing discrete logs in subgroups of prime order.

Consider the example we described above. On input a security parameter 1^n, we generate an n-bit prime q, a $poly(n)$-prime p, such that $q|p-1$ and an element g of order q in Z_p^* (the multiplicative group of integers mod$p$). In the group generated by g we can consider the exponentiation function that maps $w \in Z_q$ to $y = g^w \bmod p$. The discrete log assumption says that if we choose w at random then it is infeasible to compute w, when given only y.

Assumption 1 *We assume that computing discrete logarithm is hard, i.e. for every PPT Turing Machine* $\mathcal{I}$ *(for inverter) the following probability*

$$\mathsf{adv}_{\mathcal{I}}(n) = Prob[\mathcal{I}(p, q, g, y = g^w \bmod p) = w]$$

is negligible in n*. The probability is taken over the internal coin tosses of* $\mathcal{I}$*, the random choices of* q *as* n*-bit prime,* p *as a* $poly(n)$*-bit prime such that* $q|p-1$*,* $w \in_R Z_q$*, while* g *is an arbitrary element of order* q *in* Z_p^**.*

In the following we are going to use a stronger variant of the discrete log assumption, introduced in [BNPS01,BPa02]. In this variant once we have selected p, q at random and chose g, we give the inverter $\mathcal{I}$ access to two oracles. The *challenge* oracle Ch, when invoked outputs a random element in the group generated by g. The *discrete log* oracle DL when queried on a value $y \in (g)$ will output

w such that $y = g^w \bmod p$. The goal of $\mathcal{I}$ is to invert *all* the values issued to him by the challenge oracle (and $\mathcal{I}$ must invoke Ch at least once), but is restricted to invoke DL a number of times, which is strictly smaller than the number of times he invoked Ch. We denote with $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(n)$ the probability (taken over the choices of p, q and the internal coin tosses of $\mathcal{I}$ and Ch) that $\mathcal{I}$ succeeds in this game.

Assumption 2 *We assume that the problem of* one more inversion *of discrete logarithms is hard, i.e. we assume that* $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(n)$ *is negligible in* n.

Note that when the number of queries to Ch is equal to 1, this assumption is equivalent to Assumption 1. Though Assumption 2 is new, it looks reasonable and has the advantage enunciated in [BPa02] of reducing the security of our identification scheme to the hardness of a well-specified number theoretic problem.

2.4 Schnorr's identification scheme

Let p and q be two primes such that $q|p-1$ and $|q| = n$. Let $g \neq 1$ be an element of order q in Z_p^*. Let G_q be the subgroup generated by g. The integers p, q, g are known and can be common to a group of users.

An identity consists of a private/public key pair. The private key w is a random non-negative integer less than q. The public key is computed as $y = g^{-w} \bmod p$.

The protocol is described in Figure 1

It is well known that $\mathsf{Schnorr}$ is an honest-verifier zero-knowledge proof of knowledge of w, the discrete logarithm of y. The reader is referred to [Sch91] for details. The protocol is only honest-verifier ZK, because if a dishonest verifier chooses the challenge e in a non-random way (particularly dependent on the first message x) we are not able to simulate the interaction[2].

However in [BPa02] it is shown that the Schnorr scheme is secure against impersonation, under concurrent attacks, under the assumption that discrete logarithm is secure under one more inversion in the underlying group.

3 The new identification scheme

In this section we present our generalization of Schnorr's scheme to the case in which the prover wants to prove multiple identities.

A naive generalization of Schnorr's scheme would be to do the simultaneous authentication of d identities by composing d rounds in parallel. In other words

[2] Note that it is also necessary to check that y is in the proper group, by checking that $y^q = 1 \bmod p$, see [Bur90]. This can be added as a verification step, or the verifier can trust the certification authority that certified y to have performed the test. A similar requirement holds for our protocol.

Schnorr

Common Input: p, q, g, y. A security parameter t.
Secret Input for the Prover: $w \in Z_q$ such that $y = g^{-w} \bmod p$.

1. *Commitment by Prover.* Prover picks $r \in_R Z_q$ and sends $x = g^r \bmod p$ to the Verifier.

$$\text{Prover} \xrightarrow{\quad x = g^r \quad} \text{Verifier}$$

2. *Challenge from Verifier.* Verifier picks a number $e \in_R [1..2^t]$ and sends it to the Prover.

$$\text{Prover} \xleftarrow{\quad e \quad} \text{Verifier}$$

3. *Response from Prover.* Prover computes $s = r + w \cdot e \bmod q$ and sends it to the Verifier.

$$\text{Prover} \xrightarrow{\quad s = r + w \cdot e \quad} \text{Verifier}$$

The Verifier checks that $x = g^s \cdot y^e \bmod p$ and accepts if and only if equality holds.

Fig. 1. Schnorr's protocol

the prover would send over d commitments and the verifier would reply with d challenges - one per identity. Note that this scheme has a communication and computation cost that is d times the cost of Schnorr's original scheme. A possible improvement would be to use the same challenge for all rounds, and apply batch verification techniques (such as the ones in [BGR98]) to the last verification step. Even with these improvements, the communication and computation cost of the whole scheme would still be higher by a factor of d (the prover would still have to send and compute d commitments).

We propose a more efficient scheme where the prover sends *one* commitment and the verifier sends one challenge across all identities. The prover's response is generalized from a degree one polynomial to a degree d polynomial formed from the d secret keys. We are able to show that the resulting scheme is sound and further that it is secure against impersonation under concurrent attacks by extending the corresponding arguments in [Sch91] and [BPa02] respectively. We present two theorems that demonstrate that the new scheme is an honest-verifier zero knowledge proof of knowledge and also a secure identification against impersonation under concurrent attacks.

The parameters are very similar to Schnorr. Let p and q be two primes such that $q|p-1$. Let $g \neq 1$ be an element of order q in Z_p^*. The integers p, q, g are public, and can be common to a group of users.

We have d identities, each consisting of a private/public key pair indexed by i. The private keys w_i are non-negative integers less than q, chosen uniformly at random. The public keys are computed as $y_i = g^{-w_i} \bmod p$.

The Prover initiates the protocol by sending over the list of public keys y_i for which it claims to possess the corresponding private keys w_i. The protocol is described in Figure 2

Batch-Schnorr

Common Input: $p, q, g, y_1, \ldots, y_d$. A security parameter t.
Secret Input for the Prover: $w_i \in Z_q$ such that $y_i = g^{-w_i} \bmod p$.

1. *Commitment by Prover.* Prover picks $r \in_R Z_q$ and sends $x = g^r \bmod p$ to the Verifier.

$$\text{Prover} \xrightarrow{\quad x = g^r \quad} \text{Verifier}$$

2. *Challenge from Verifier.* Verifier picks a number $e \in_R [1..2^{(t+\log d)}]$ and sends it to the Prover.

$$\text{Prover} \xleftarrow{\quad e \quad} \text{Verifier}$$

3. *Response from Prover.* Prover computes $s = r + \Sigma_i w_i \cdot e^i \bmod q$ and sends it to the Verifier.

$$\text{Prover} \xrightarrow{\quad s = r + \Sigma_i w_i \cdot e^i \quad} \text{Verifier}$$

The Verifier checks that $x = g^y \cdot \Pi_i y_i^{e^i} \bmod p$, and accepts if and only if equality holds.

Fig. 2. Batch version of Schnorr's protocol

Theorem 1. Batch-Schnorr *is an honest-verifier zero-knowledge proof of knowledge for d discrete logarithms.*

The complete proof is provided in [GLSY].

Notice that the protocol is not zero-knowledge in the general case since a dishonest verifier could choose a challenge that is dependent on the commitment making it difficult to generate transcripts with the same distribution, without knowing the secret keys. Informally, however the reason no information is revealed is that the numbers x and y, the commitment and the response, are essentially random. This is the intuition behind the proof of security as an identification scheme.

The following theorem (Theorem 2) shows that Batch-Schnorr is an identification scheme secure against impersonation under concurrent attacks. As mentioned before, this is our ultimate end goal. We extend the proof in [BPa02] (which shows the security of Schnorr's scheme under this kind of attack) to our scheme. We remind the readers that in a concurrent attack the adversarial prover is allowed to play the role of the cheating verifier and interact concurrently with multiple honest prover clones prior to the impersonation attempt. Similar to [BPa02] our proof is based on the assumption that discrete exponentiation is secure under d more inversions in the underlying group (Assumption 2).

Let $\mathcal{A}$ denote the adversary that first takes on the role of fraudulent verifier and interacts concurrently with several honest prover clones before subsequently taking on the role of fraudulent prover. Let $\mathsf{adv}_{\mathcal{A}}(k)$ denote the probability that $\mathcal{A}$ is successful at impersonation.

Theorem 2. *If $\mathcal{A}$ succeeds in an impersonation attack on* Batch-Schnorr *with probability $\mathsf{adv}_{\mathcal{A}}(k)$ then there exists an inverter $\mathcal{I}$ such that for every k*

$$\mathsf{adv}_{\mathcal{A}}(k) \leq 2^{-t} + (\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch},\mathsf{DL}}(k))^{1/(d+1)}$$

Proof. We show how to construct an inverter $\mathcal{I}$ that interacts with $\mathcal{A}$ the impersonation adversary. Via this interaction $\mathcal{I}$ will compute the discrete logarithm of all the n points it gets from the challenge oracle, by querying the discrete log oracle, at most $n-d$ times.

First the inverter $\mathcal{I}$ queries Ch, the challenge oracle, d times and obtains d random group elements $y_i = g^{-w_i}$. It then runs $\mathcal{A}$ in cheating verifier mode using the y_i's as the public key. For the j^{th} clone prover the commitment x_j is obtained by querying the challenge oracle Ch. The third round response s_j to challenge e_j is computed by querying the discrete log oracle DL on the value $x_j \Pi_{k=1}^{d} y_k^{-e_j^k}$. Notice that this is a perfect simulation of a real concurrent attack. With n we denote the total number of queries to the challenge oracle. Notice that $\mathcal{I}$ queried the discrete log oracle only $n-d$ times.

Now $\mathcal{I}$ runs $\mathcal{A}$ in cheating prover mode $d+1$ times, rewinding it each time to the beginning (of the phase in which $\mathcal{A}$ acts as a prover). This in particular means that the commitment issued by $\mathcal{A}$ stays the same, since its internal state is the same. If any two challenges are the same then the inverter $\mathcal{I}$ fails.

Let $x = g^r$ be the commitment and let s_i be the response corresponding to the distinct challenges e_i. If the cheating prover $\mathcal{A}$ fails even once then the inverter $\mathcal{I}$ fails. If the cheating prover $\mathcal{A}$ succeeds each of the $d+1$ times, then the inverter $\mathcal{I}$ has $d+1$ equations of the form $s_i = r + \Sigma_j w_j \cdot e_i^j$, with $d+1$ unknowns, r and the d secret keys w_j. By inverting the Van der Monde matrix formed from these equations, they can be solved to obtain the w_j's. These are the answers to the first d queries $\mathcal{I}$ made to Ch.

Recall that $\mathcal{I}$ must answer *all* the challenges he received from Ch. But the answer to each query x_j can be easily computed as $s_j - \Sigma_k w_k \cdot e_j^k$.

Thus with $n-d$ queries, $\mathcal{I}$ succeeds in inverting all the n points asked to the challenge oracle. The probability of success is the probability that $\mathcal{A}$ succeeds $d+1$ times. We will now estimate this probability. We first prove an auxiliary result that is a generalization of an equivalent result in [BPa02].

Lemma 1 (Generalized Reset Lemma). *Consider any prover (potentially cheating). Let A and B be random variables over the space of the random coins, RP, of the prover. Let A denote the probability, taken over e, that the verifier accepts. Let B denote the probability, taken over e's, that when the verifier is reset and run $d+1$ times, a different e is generated each time and the verifier accepts each time. Let $acc = E(A)$ and $res = E(B)$. Then $acc \leq 2^{-t} + res^{1/(d+1)}$.*

Proof. Let $1/c = 2^{t+\log d}$ be the size of the challenge set i.e. $\#e$. It is easy to see that $B \geq A(A-c)(A-2c)\dots(A-dc)$. This implies that $B \geq (A-dc)^{(d+1)}$ which yields that $E(A) \leq dc + E(B)^{1/(d+1)}$ or $E(A) \leq 2^{-t} + E(B)^{1/d+1}$.

Now observe that $\mathsf{adv}_{\mathcal{A}}(k) = E(acc)$, where the expectation is taken over the choice of y_i and the knowledge gained as the cheating verifier. Similarly $\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch,DL}}(k) = E(res)$. Applying the reset lemma we see that

$$adv_{\mathcal{A}}(k) = E(acc) \leq E(2^{-t} + (res)^{1/(d+1)}) = 2^{-t} + E((res)^{1/(d+1)})$$

then by applying Jensen's inequality

$$adv_{\mathcal{A}}(k) \leq 2^{-t} + (E(res))^{1/(d+1)} = 2^{-t} + (\mathsf{adv}_{\mathcal{I}}^{\mathsf{Ch,DL}})^{1/(d+1)}$$

This completes the proof of Theorem 2.

The following corollary is a straightforward consequence of Theorem 2.

Corollary 1. *Under Assumption 2, and if $t = \omega(\log k)$, then* Batch-Schnorr *is a secure identification scheme against i mpersonation under concurrent attack.*

Note that the assumption that t is super-logarithmic in k is necessary, otherwise the scheme can be broken by guessing the verifier's challenge.

3.1 Efficiency Analysis

For a list of d identities Batch-Schnorr uses only $O(logd)$ more bits of communication than Schnorr's scheme for a single identity (assuming the same security level).

In terms of computation Batch-Schnorr requires $2d$ extra modular multiplications for the prover. The verifier has to perform $d+1$ modular exponentiations, while in Schnorr's scheme it has to perform 2.

Notice that this is much faster than the known way of proving d identities simultaneously, which consists of d copies of Schnorr's protocol (in the particular the verifier would have to perform $2d$ exponentiations instead of $d+1$).

3.2 Authorization using Multiple Identities

As we discussed in the Introduction, our identification scheme is suitable to implement Authorization using multiple identities without incurring a huge efficiency cost.

When a user joins a particular privilege class he is given a new public key, its matching secret key and a certificate that associates the key to that particular privilege class. Another possibility would be to have a unique key for each class, but that would make revocation very difficult to handle, as revoking one user in the class (say because her key was compromised or because she does not belong to the class anymore), would involve replacing the key of all users in the class.

When a user needs to access certain data or services he uses our identification protocol to prove possession of the minimal set of privileges required for that access to take place.

Another advantage of our approach is that it is easy to transfer privileges among parties. A motivating scenario for our application is having a smart card be able to talk to another smart card and transfer a subset of privileges to it (equivalent to a high ranking employee in a corporation enabling selective access to a lower ranked employee). In our scheme using multiple identities it is a simple matter to transfer the private keys corresponding to the selected list of privileges.

4 Implementation

In order to test the efficiency of our scheme we have performed an implementation of our scheme. To carry out the implementation we used a recently proposed technology based on Light Emitting Diodes. We believe that another contribution of our paper is to show that this new technology allows the implementation of strong cryptography.

Light Emitting Diodes, or LEDs, are one of the most ubiquitous interface components. Their diverse applications include numeric displays, flashlights, vehicle brake lights (and possibly even headlights [Hel03]), traffic signals and the omni-present power-on indicator. LEDs are so commonly used as light emitters that people often forget that they are fundamentally photodiodes and hence light *detectors*. Although LEDs are not optimized for light detection they are very effective at it. The interchangeability between solid-state light emission and detection was widely publicized in the 1970s by Forrest W. Mims [Mim86,Mim93], but has since largely been forgotten.

Recently, a novel microprocessor interface circuit was invented which can alternately emit and detect light using an LED [DYL02]. In addition to the LED and two digital I/O pins of the microprocessor, the circuit requires only a single current limiting resistor. When forward-biased the LED emits light and when back-biased it detects/measures the ambient light. The implications of LED-based data communication are significant, since it is essentially a software interface technique that uses existing hardware with minimal modification. "Every LED connected to a microprocessor can be thought of as a generic two-way data port" [DYL02]. One can conceive of numerous applications e.g. using the power light on consumer appliances as a maintenance port for reading service information and uploading new firmware, or capturing a car stereo's fault log through the front panel display.

We show how to build smartcards that communicate via LEDs and implement our Batch-Schnorr protocol.

4.1 Hardware

Batch-Schnorr was implemented using the Microchip PIC16LF628 microcontroller. The hardware was composed of a small printed circuit board 2cm by 4cm, a single push-button switch, an LED, a 3-volt lithium coin-cell battery, a capacitor and two resistors. The PIC uses 8-bit instruction words and runs at 5 MIPS (million instructions per second). It has 16KB of write-able storage. The prototypes were also equipped with an in-circuit programming connector, which allowed us to download code into the microcontroller. We also devised a small adapter board to convert this connector to Microchip's standard RJ-11 in-circuit debugging module. A mass produced version should cost less than a dollar more than a similar LED keychain flashlight. The range of communication is a few centimeters at best and the data rate is 250 bits/second in each direction. We implemented Batch-Schnorr representing the prover in its full functionality. The verifier was implemented as a LED directly controlled by a PC.

See Appendix B for a picture of our implementation.

4.2 Security

We chose a security parameter setting of $t = 95$. This is generally considered adequate security (see [Sch96]) for most practical purposes. We used $d = 32$. This made $t + \log d = 100$. This forced the prime q to be 200 bits long because of the existence of the $O(q^{1/2})$ baby-step-giant-step algorithm for finding discrete logs (see [Sch91]). In conjunction with the existence of the general number field sieve (see [LOd91]) this, in turn, forced the prime p to be about 1500 bits long.

4.3 Prover

The bulk of the implementation effort lay in the code for the prover. An important aspect of our implementation of Batch-Schnorr was that storage was at a premium. This is common with most smart cards where the storage is needed both for code as well as data.

The main operation performed by the prover is modular multiplication. We initially attempted an implementation of the Fast Fourier Transform (see [Str88]) of Cooley and Tukey, which takes $O(n \log n)$ bit operations. However it turned out that our practical implementations of this scheme had high code complexity, even though it is more efficient asymptotically.

Hence, we adopted a scheme that utilizes a pre-computed table to substantially save on both code complexity as well as computation time. For each of the private keys we stored a pre-computed table of the residues modulo q of the product of the private key with the powers of 2 up to $2^{1+\log q}$. Then to multiply the private key with any given number we added the residues corresponding to the powers of 2 present in the binary representation of that number. The residue modulo q of $2^{1+\log q}$ enabled us to reduce the overflow when doing addition, so that we always had a number with $\log q$ bits. Upon receiving the challenge e we first computed a similar table consisting of the residues modulo q of the product of e with the powers of 2 up to $2^{1+\log q}$. We then used this table to compute the powers of e, and then used the pre-computed tables of the secret keys to compute $y = r + \Sigma_i si \cdot e^i \bmod q$. This enhancement enabled the implementation to run in less than 2 seconds for our choice of the security parameters.

5 Conclusion and Extensions

We have presented a batch version of Schnorr's protocol. In our scheme a prover can prove knowledge of d keys at essentially the same cost as proving knowledge of a single key. We believe this protocol can find several applications in the cryptography literature.

We discussed the application of privacy-preserving authorization mechanisms. Also we presented an implementation of our protocol employing a new technology to use Light Emitting Diodes as two-way communication devices. We believe this to be another interesting contribution of our paper.

In terms of future research, it would be interesting to devise a batch version of the Guillou-Quisquateur identification protocol.

References

[BGR98] M. Bellare, J. Garay and T. Rabin. "Fast batch verification for modular exponentiation and digital signatures". Advances in Cryptology- Eurocrypt '98 Proceedings, Lecture Notes in Computer Science Vol. 1403, K. Nyberg ed, Springer-Verlag, 1998.

[BGo93] M. Bellare and O. Goldreich. "On defining proofs of knowledge". Advances in Cryptology - CRYPTO '92 Proceedings, Lecture Notes in Computer Science Vol. 740, E. Brickell ed, Springer-Verlag, 1993.

[BNPS01] M. Bellare, C. Namprempre, D. Pointcheval and M. Semanko. *The one-more RSA Inversion Problem.* Financial Cryptography'01. Final version available at `http://eprint.iacr.org/2002/002`

[BPa02] Bellare M., and Palacio A., "GQ and Schnorr identification schemes: proofs of security against impersonation under active and concurrent attacks," Advances in Cryptology-CRYPTO '02, (2002).

[Bur90] Burmester M., "A remark on the efficiency of identification schemes," EUROCRYPT '90, pp. 493-495 (1990).

[CL01] J. Camenisch and A. Lysyanskaya. "An efficient system for non-transferable anonymous credentials with optional anonymity revocation." EUROCRYPT'01 pp.93-118.

[CL02] J. Camenisch and A. Lysyanskaya. "Signature schemes with efficient protocols." Security in Communication Networks Workshop. 2002

[Cha90] Chaum D., "Zero knowledge undeniable signatures," Advances in Cryptology - CRYPTO '90, pp 458-464 (1990).

[CDM00] Cramer R., Damgard I., and MacKenzie P.D., "Efficient zero-knowledge proofs of knowledge without intractability assumptions," Public Key Cryptography '00, pp. 354-372 (2002).

[CSh98] Cramer R., and Shoup V., "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," Advances in Cryptology-CRYPTO '98, pp. 13-25 (1998).

[CSh99] R.Cramer and V.Shoup. "Signature schemes based on the Strong RSA assumption". 6th ACM Conference on Computer and Communication Security 1999.

[DPr89] Davies D. W., and Price W. L., "Security for computer networks," Wiley (1989).

[deR91] de Rooij P., "On the security of the Schnorr scheme using preprocessing," EUROCRYPT '91, pp. 71-80 (1991).

[DYL02] Dietz, P., Yerazunis W., and Leigh, D., "Very low-cost sensing and communication using bidirectional LEDs," to appear in Ubicomp 2003. http://www.merl.com/papers/TR2003-35/ (2002) patent pending.

[DH78] W. Diffie and M. Hellman. *New Directions in Cryptography.* IEEE Transactions on Information Theory, IT-22(6):644–654. 1976.

[DDN00] D.Dolev, C.Dwork and M.Naor. "Non-malleable Cryptography". SIAM J. Comp. 30(2):391–437, 2000.

[FFS88] Feige U., Fiat A., and Shamir A., "Zero-knowledge proofs of identity," Journal of Cryptology, vol. 1, pp. 77-94 (1988).

[For94] Ford W., "Computer communications security: principles, standard protocols and techniques," Prentice Hall (1994).

[GHR99] Gennaro R., Halevi S. and Rabin T., "Secure Hash-and-Sign Signatures Without the Random Oracle". Eurocrypt '99 LNCS no. 1592, pages 123-139 (1999).

[GKR97] Gennaro R., Krawczyk H. and Rabin T., "RSA-based undeniable signatures," Advances in Cryptography - CRYPTO '97, pp 132-149 (1997). Also in Journal of Cryptology, vol 13., pp 397-416 (2000).

[GLSY] Gennaro R., Leigh D., Sundaram R. and Yerazunis W., "Batching Schnorr Identification Scheme with Applications to Privacy-Preserving Authorization and Low-Bandwidth Communication Devices," Tech Report. http://www.ccs.neu.edu/ koods/papers.html (2004).

[GMR85] Goldwasser S., Micali S., and Rackoff C., "The knowledge complexity of interactive proof-systems," Proceedings of the 17th Annual ACM Symposium on Theory of Computing, pp. 291-304 (1985).

[GMR88] S.Goldwasser, S.Micali, and R.L.Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". *SIAM J. Computing*, 17(2):281–308, April 1988.

[GQu88] Guillou L. C., and Quisquater J. -J., "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory," EUROCRYPT '88, pp. 123-128 (1988).

[GUg86] Guillou L. C., and Ugon M., "Smart card: a highly reliable and portable security device," Advances in Cryptology-CRYPTO '86, pp. 464-479 (1986).

[GUQ92] Guillou L. C., Ugon M., and Quisquater J. -J., "The smart card: a standardized security device dedicated to public cryptography," Contemporary Cryptology: The Science of Information Integrity, IEEE, pp. 561-613 (1992).

[Hel03] Helms N., "Bright LEDs power headlights," Electronics News, http://www.dialelectronics.com.au/articles/1e/0c01631e.asp (2003).
[Kah67] Kahn D., "The codebreakers: the story of secret writing," Macmillan (1967).
[LOd91] LaMacchia B. A., and Odlyzko A. M., "Computation of discrete logarithms in prime fields," Designs, Codes and Cryptography, vol. 1, pp. 46-62 (1991).
[Mim86] Mims F. M., "Siliconnections: Coming of age in the electronic era," McGraw Hill (1986).
[Mim93] Mims F. M., "LED circuits and projects," H. W. Sams and Co. (1993).
[MOV97] Menezes A. J., van Oorschot P. C., and Vanstone S. A., "Handbook of applied cryptography," CRC Press (1997). http://www.cacr.math.uwaterloo.ca/hac/
[MTh79] Morris R., and Thompson K., "Password security: a case history," Communications of the ACM, vol. 22, pp. 594-597 (1979).
[NSc78] Needham R. M., and Schroeder M. D., "Using encryption for authentication in large networks of computers," Communications of the ACM, vol. 21, pp. 993-999 (1978).
[QGB89] Quisquater J. -J., Guillou L., and Berson T., "How to explain zero-knowledge protocols to your children," Advances in Cryptology-CRYPTO '89, LNCS 435, pp. 628-631 (1989).
[Sch96] Schneier B., "Applied cryptography," Wiley (1996).
[Sch91] Schnorr C. P., "Efficient signature generation for smart cards," Journal of Cryptology, vol. 4, no. 3, pp. 161-174 (1991).
[Str88] Strang G., "Linear algebra and its applications," Harcourt Brace (1988).
[Ted02] Tedeschi, W., "Trying to shift shape of PC screens," http://www.nytimes.com/2002/11/04/technology/04ECOM.html

A LightKey Image

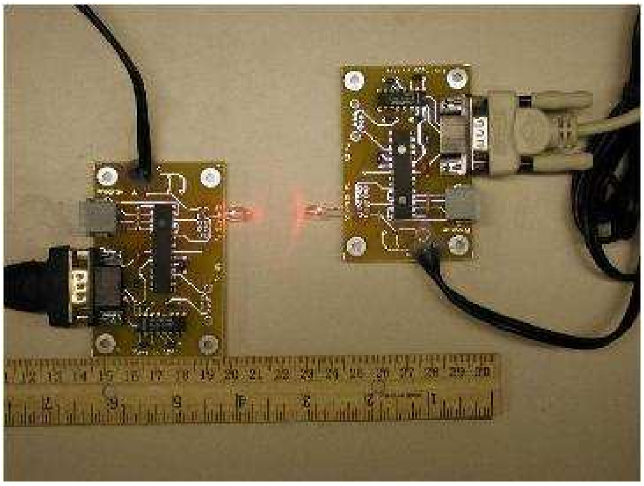

Fig. 3. LightKey Image