CHAPTER 21

Identification Schemes

21.1 FEIGE-FIAT-SHAMIR

Amos Fiat's and Adi Shamir's authentication and digital signature scheme is discussed in [566,567]. Uriel Feige, Fiat, and Shamir modified the algorithm to a zero-knowledge proof of identity [544,545]. This is the best-known zero-knowledge proof of identity.

On July 9, 1986 the three authors submitted a U.S. patent application [1427]. Because of its potential military applications, the application was reviewed by the military. Occasionally the Patent Office responds not with a patent, but with something called a secrecy order. On January 6, 1987, three days before the end of their six-month period, the Patent Office imposed that order at the request of the Army. They stated that ". . . the disclosure or publication of the subject matter . . . would be detrimental to the national security. . . ." The authors were ordered to notify all Americans to whom the research had been disclosed that unauthorized disclosure could lead to two years' imprisonment, a $10,000 fine, or both. Furthermore, the authors had to inform the Commissioner of Patents and Trademarks of all foreign citizens to whom the information had been disclosed.

This was ludicrous. All through the second half of 1986, the authors had presented the work at conferences throughout Israel, Europe, and the United States. The authors weren't even American citizens, and all the work had been done at the Weizmann Institute in Israel.

Word spread through the academic community and the press. Within two days the secrecy order was rescinded; Shamir and others believe that the NSA pulled strings to rescind the order, although they officially had no comment. Further details of this bizarre story are in [936].

Simplified Feige-Fiat-Shamir Identification Scheme

Before issuing any private keys, the arbitrator chooses a random modulus, n, which is the product of two large primes. In real life, n should be at least 512 bits

long and probably closer to 1024 bits. This n can be shared among a group of provers. (Choosing a Blum integer makes computation easier, but it is not required for security.)

To generate Peggy's public and private keys, a trusted arbitrator chooses a number, v, where v is a quadratic residue mod n. In other words, choose v such that $x^2 \equiv v \pmod{n}$ has a solution and $v^{-1} \bmod n$ exists. This v is Peggy's public key. Then calculate the smallest s for which $s \equiv \text{sqrt}(v^{-1}) \pmod{n}$. This is Peggy's private key.

The identification protocol can now proceed.

(1) Peggy picks a random r, where r is less then n. She then computes $x = r^2 \bmod n$, and sends x to Victor.

(2) Victor sends Peggy a random bit, b.

(3) If $b = 0$, then Peggy sends Victor r. If $b = 1$, then Peggy sends Victor $y = r * s \bmod n$.

(4) If $b = 0$, Victor verifies that $x = r^2 \bmod n$, proving that Peggy knows sqrt (x). If $b = 1$, Victor verifies that $x = y^2 * v \bmod n$, proving that Peggy knows sqrt (v^{-1}).

This is a single round—called an **accreditation**—of the protocol. Peggy and Victor repeat this protocol t times, until Victor is convinced that Peggy knows s. It's a cut-and-choose protocol. If Peggy doesn't know s, she can pick r such that she can fool Victor if he sends her a 0, or she can pick r such that she can fool Victor if he sends her a 1. She can't do both. The odds of her fooling Victor once are 50 percent. The odds of her fooling him t times are 1 in 2^t.

Another way for Victor to attack the protocol would be trying to impersonate Peggy. He could initiate the protocol with another verifier, Valerie. In step (1), instead of choosing a random r, he would just reuse an old r that he saw Peggy use. However, the odds of Valerie choosing the same value for b in step (2) that Victor did in the protocol with Peggy are 1 in 2. So, the odds of his fooling Valerie are 50 percent. The odds of his fooling her t times are 1 in 2^t.

For this to work, Peggy must not reuse an r, ever. If she did, and Victor sent Peggy the other random bit in step (2), then he would have both of Peggy's responses. Then, from even one of these, he can calculate s and it's all over for Peggy.

Feige-Fiat-Shamir Identification Scheme

In their papers [544,545], Feige, Fiat and Shamir show how parallel construction can increase the number of accreditations per round and reduce Peggy and Victor's interactions.

First generate n as in the previous example, the product of two large primes. To generate Peggy's public and private keys, first choose k different numbers: v_1, v_2, . . . , v_k, where each v_i is a quadratic residue mod n. In other words, choose v_i such that $x^2 = v_i \bmod n$ has a solution and $v_i^{-1} \bmod n$ exists. This string, v_1, v_2, . . . , v_k, is the public key. Then calculate the smallest s_i such that $s_i = \text{sqrt}(v_i^{-1}) \bmod n$. This string, s_1, s_2, . . . , s_k, is the private key.

And the protocol is:

(1) Peggy picks a random r, when r is less than n. She then computes $x = r^2 \bmod n$, and sends x to Victor.

(2) Victor sends Peggy a random binary string k-bits long: $b_1, b_2, \ldots, b_k$.

(3) Peggy computes $y = r * (s_1^{b_1} * s_2^{b_2} * \ldots * s_k^{b_k}) \bmod n$. (She multiplies together whichever values of s_i that correspond to $b_i = 1$. If Victor's first bit is a 1, then s_1 is part of the product; if Victor's first bit is a 0, then s_1 is not part of the product, and so on.) She sends y to Victor.

(4) Victor verifies that $x = y^2 * (v_1^{b_1} * v_2^{b_2} * \ldots * v_k^{b_k}) \bmod n$. (He multiplies together the values of v_i based on the random binary string. If his first bit is a 1, then v_1 is part of the product; if his first bit is a 0, then v_1 is not part of the product, and so on.)

Peggy and Victor repeat this protocol t times, until Victor is convinced that Peggy knows $s_1, s_2, \ldots, s_k$.

The chance that Peggy can fool Victor is 1 in 2^{kt}. The authors recommend a 1 in 2^{20} chance of a cheater fooling Victor and suggest that $k = 5$ and $t = 4$. If you are more paranoid, increase these numbers.

An Example

Let's look at this protocol in action with small numbers.

If $n = 35$ (the two primes are 5 and 7), then the possible quadratic residues are:

1: $x^2 \equiv 1 \pmod{35}$ has the solutions: $x = 1, 6, 29$, or 34.
4: $x^2 \equiv 4 \pmod{35}$ has the solutions: $x = 2, 12, 23$, or 33.
9: $x^2 \equiv 9 \pmod{35}$ has the solutions: $x = 3, 17, 18$, or 32.
11: $x^2 \equiv 11 \pmod{35}$ has the solutions: $x = 9, 16, 19$, or 26.
14: $x^2 \equiv 14 \pmod{35}$ has the solutions: $x = 7$ or 28.
15: $x^2 \equiv 15 \pmod{35}$ has the solutions: $x = 15$ or 20.
16: $x^2 \equiv 16 \pmod{35}$ has the solutions: $x = 4, 11, 24$, or 31.
21: $x^2 \equiv 21 \pmod{35}$ has the solutions: $x = 14$ or 21.
25: $x^2 \equiv 25 \pmod{35}$ has the solutions: $x = 5$ or 30.
29: $x^2 \equiv 29 \pmod{35}$ has the solutions: $x = 8, 13, 22$ or 27.
30: $x^2 \equiv 30 \pmod{35}$ has the solutions: $x = 10$ or 25.

The inverses (mod 35) and their square roots are:

v	v^{-1}	$s = \text{sqrt}(v^{-1})$
1	1	1
4	9	3
9	4	2

11	16	4
16	11	9
29	29	8

Note that 14, 15, 21, 25, and 30 do not have inverses mod 35, because they are not relatively prime to 35. This makes sense, because there should be $(5 - 1) * (7 - 1)/4$ quadratic residues mod 35 relatively prime to 35: That is $\gcd(x,35) = 1$ (see Section 11.3).

So, Peggy gets the public key consisting of $k = 4$ values: {4,11,16,29}. The corresponding private key is {3,4,9,8}. Here's one round of the protocol.

(1) Peggy chooses a random $r = 16$, computes $16^2 \bmod 35 = 11$, and sends it to Victor.

(2) Victor sends Peggy a random binary string {1,1,0,1}.

(3) Peggy computes $16 * ((3^1) * (4^1) * (9^0) * (8^1)) \bmod 35 = 31$ and sends it to Victor.

(4) Victor verifies that $31^2 * ((4^1) * (11^1) * (16^0) * (29^1)) \bmod 35 = 11$.

Peggy and Victor repeat the protocol t times, each time with a different random r, until Victor is satisfied.

With small values like these, there's no real security. But when n is 512 bits long or more, Victor cannot learn anything about Peggy's secret key except the fact that she knows it.

Enhancements

It is possible to embed identification information into the protocol. Assume that I is a binary string representing Peggy's identification: her name, address, social security number, hat size, preferred brand of soft drink, and other personal information. Use a one-way hash function $H(x)$ to compute $H(I,j)$, where j is a small number concatenated onto I. Find a set of js where $H(I,j)$ is a quadratic residue mod n. These $H(I,j)$s become $v_1, v_2, \ldots, v_k$ (the js need not be quadratic residues). Peggy's public key is now I and the list of js. She sends I and the list of js to Victor before step (1) of the protocol (or perhaps Victor downloads them from a public bulletin board someplace), and Victor generates $v_1, v_2, \ldots, v_k$ from $H(I,j)$.

Now, after Victor successfully completes the protocol with Peggy, he is assured that Trent, who knows the factorization of the modulus, has certified the association between I and Peggy by giving her the square roots of the v_i derived from I. (See Section 5.2 for background information.)

Feige, Fiat, and Shamir include the following implementation remarks [544,545]:

> For nonperfect hash functions, it may be advisable to randomize I by concatenating it with a long random string, R. This string is chosen by the arbitrator and is revealed to Victor along with I.
>
> In typical implementations, k should be between 1 and 18. Larger values of k can reduce the time and communication complexity by reducing the number of rounds.

The value n should be at least 512 bits long. (Of course, there has been considerable progress in factoring since then.)

If each user chooses his own n and publishes it in a public key file, they can dispense with the arbitrator. However, this RSA-like variant makes the scheme considerably less convenient.

Fiat-Shamir Signature Scheme

Turning this identification scheme into a signature scheme is basically a matter of turning Victor into a hash function. The primary benefit of the Fiat-Shamir digital signature scheme over RSA is speed: Fiat-Shamir requires only 1 percent to 4 percent of the modular multiplications of RSA. For this protocol, we'll bring back Alice and Bob.

The setup is the same as the identification scheme. Choose n to be the product of two large primes. Generate the public key, $v_1, v_2, \ldots, v_k$, and the private key, $s_1, s_2, \ldots, s_k$, such that $s_i = \text{sqrt}\,(v_i^{-1}) \bmod n$.

(1) Alice picks t random integers between 1 and n: $r_1, r_2, \ldots, r_t$, and computes $x_1, x_2, \ldots, x_t$ such that $x_i = r_i^2 \bmod n$.

(2) Alice hashes the concatenation of the message and the string of x_is to generate a bit stream: $H(m, x_1, x_2, \ldots, x_t)$. She uses the first $k * t$ bits of this string as values of b_{ij}, where i goes from 1 to t, and j goes from 1 to k.

(3) Alice computes $y_1, y_2, \ldots, y_t$, where

$$y_i = r_i * (s_1^{b_{i1}} * s_2^{b_{i2}} * \ldots * s_k^{b_{ik}}) \bmod n$$

(For each i, she multiplies together the values of the s_j based on the random $b_{i,j}$ values. If $b_{i,1}$ is a 1, then s_1 is multiplied; if $b_{i,1}$ is a 0, then s_1 is not multiplied.)

(4) Alice sends Bob m, all the bit values of $b_{i,j}$, and all the values of y_i. He already has Alice's public key: $v_1, v_2, \ldots, v_k$.

(5) Bob computes $z_1, z_2, \ldots, z_t$, where

$$z_i = y_i^2 * (v_1^{b_{i1}} * v_2^{b_{i2}} * \ldots * v_k^{b_{ik}}) \bmod n$$

(Again, Bob multiplies based on the $b_{i,j}$ values.) Also note that z_i should be equal to x_i.

(6) Bob verifies that the first $k * t$ bits of $H(m, z_1, z_2, \ldots, z_t)$ are the $b_{i,j}$ values that Alice sent him.

As with the identification scheme, the security of this signature scheme is proportional to $1/2^{kt}$. It also depends on the difficulty of factoring n. Fiat and Shamir pointed out that forging a signature is easier when the complexity of factoring n is considerably lower than 2^{kt}. And, because of birthday-type attacks (see Section 18.1), they recommend that $k * t$ be increased from 20 to at least 72. They suggest $k = 9$ and $t = 8$.

Improved Fiat-Shamir Signature Scheme

Silvio Micali and Adi Shamir improved the Fiat-Shamir protocol in [1088]. They chose $v_1, v_2, \ldots, v_k$ to be the first k prime numbers. So

$$v_1 = 2,\ v_2 = 3,\ v_3 = 5, \text{ and so on.}$$

This is the public key.

The private key, $s_1, s_2, \ldots, s_k$ is a random square root, determined by

$$s_i = \text{sqrt}\ (v_i^{-1}) \bmod n$$

In this version, every person must have a different n. The modification makes it easier to verify signatures. The time required to generate signatures, and the security of those signatures, is unaffected.

Other Enhancements

There is also an N-party identification scheme, based on the Fiat-Shamir algorithm [264]. Two other improvements to the Fiat-Shamir scheme are proposed in [1218]. Another variant is [1368].

Ohta-Okamoto Identification Scheme

This protocol is a modification of the Feige-Fiat-Shamir identification scheme and gets its security from the difficulty of factoring [1198,1199]. The same authors also wrote a multisignature scheme (see Section 23.1), by which a number of different people can sequentially sign a message [1200]. This scheme has been proposed for smart-card implementation [850].

Patents

Fiat-Shamir is patented [1427]. Anyone interested in licensing the algorithm should contact Yeda Research and Development, The Weizmann Institute of Science, Rehovot 76100, Israel.

21.2 GUILLOU-QUISQUATER

Feige-Fiat-Shamir was the first practical identity-based protocol. It minimized computation by increasing the number of iterations and accreditations per iteration. For some implementations, like smart cards, this is less than ideal. Exchanges with the outside world are time-consuming, and the storage required for each accreditation can strain the limited resources of the card.

Louis Guillou and Jean-Jacques Quisquater developed a zero-knowledge identification algorithm more suited to applications like these [670,1280]. The exchanges between Peggy and Victor and the parallel accreditations in each exchange are both kept to an absolute minimum: There is only one exchange of one accreditation for each proof. For the same level of security, the computation required by Guillou-Quisquater is greater than by Feige-Fiat-Shamir by a factor of three. And

like Feige-Fiat-Shamir, this identification algorithm can be converted to a digital signature algorithm.

Guillou-Quisquater Identification Scheme

Peggy is a smart card who wants to prove her identity to Victor. Peggy's identity consists of a set of credentials: a data string consisting of the card's name, validity period, a bank account number, and whatever else the application warrants. This bit string is called J. (Actually, the credentials can be a longer string and hashed to a J value. This complexity does not modify the protocol in any way.) This is analogous to the public key. Other public information, shared by all "Peggys" who could use this application, is an exponent v and a modulus n, where n is the product of two secret primes. The private key is B, calculated such that $JB^v \equiv 1 \pmod{n}$.

Peggy sends Victor her credentials, J. Now, she wants to prove to Victor that those credentials are hers. To do this, she has to convince Victor that she knows B. Here's the protocol:

(1) Peggy picks a random integer r, such that r is between 1 and $n - 1$. She computes $T = r^v \bmod n$ and sends it to Victor.

(2) Victor picks a random integer, d, such that d is between zero and $v - 1$. He sends d to Peggy.

(3) Peggy computes $D = rB^d \bmod n$, and sends it to Victor.

(4) Victor computes $T' = D^v J^d \bmod n$. If $T \equiv T' \pmod{n}$, then the authentication succeeds.

The math isn't that complex:

$$T' = D^v J^d = (rB^d)^v J^d = r^v B^{dv} J^d = r^v (JB^v)^d = r^v \equiv T \pmod{n}$$

since B was constructed to satisfy

$$JB^v \equiv 1 \pmod{n}$$

Guillou-Quisquater Signature Scheme

This identification can be converted to a signature scheme, also suited for smartcard implementation [671,672].

The public and private key setup is the same as before. Here's the protocol:

(1) Alice picks a random integer r, such that r is between 1 and $n - 1$. She computes $T = r^v \bmod n$.

(2) Alice computes $d = H(M,T)$, where M is the message being signed and $H(x)$ is a one-way hash function. The d produced by the hash function must be between 0 and $v - 1$ [1280]. If the output of the hash function is not within this range, it must be reduced modulo v.

(3) Alice computes $D = rB^d \bmod n$. The signature consists of the message, M, the two calculated values, d and D, and her credentials, J. She sends this signature to Bob.

(4) Bob computes $T' = D^v J^d \bmod n$. He then computes $d' = H(M,T')$. If $d = d'$, then Alice must know B and the signature is valid.

Multiple Signatures

What if many people want to sign the same document? The easy solution has each of them signing separately, but this signature scheme can do better than that. Here Alice and Bob sign the same document and Carol verifies the signatures, but any number of people can be involved in the signature process. As before, Alice and Bob have their own unique J and B values: (J_A, B_A) and (J_B, B_B). The values n and v are common to the system.

(1) Alice picks a random integer, r_A, such that r_A is between 1 and $n - 1$. She computes $T_A = {r_A}^v \bmod n$ and sends T_A to Bob.

(2) Bob picks a random integer, r_B, such that r_B is between 1 and $n - 1$. He computes $T_B = {r_B}^v \bmod n$ and sends T_B to Alice.

(3) Alice and Bob each compute $T = (T_A T_B) \bmod n$.

(4) Alice and Bob each compute $d = H(M,T)$, where M is the message being signed and $H(x)$ is a one-way hash function. The d produced by the hash function must be between 0 and $v - 1$ [1280]. If the output of the hash function is not within this range, it must be reduced modulo v.

(5) Alice computes $D_A = r_A {B_A}^d \bmod n$ and sends D_A to Bob.

(6) Bob computes $D_B = r_B {B_B}^d \bmod n$ and sends D_B to Alice.

(7) Alice and Bob each compute $D = D_A D_B \bmod n$. The signature consists of the message, M, the two calculated values, d and D, and both of their credentials: J_A and J_B.

(8) Carol computes $J = J_A J_B \bmod n$.

(9) Carol computes $T' = D^v J^d \bmod n$. She then computes $d' = H(M,T')$. If $d \equiv d'$, then the multiple signature is valid.

This protocol can be extended to any number of people. For multiple people to sign, they all multiply their individual T_i values together in step (3), and their individual D_i values together in step (7). To verify a multiple signature, multiply all the signers J_i values together in step (8). Either all the signatures are valid or there is at least one invalid signature.

21.3 SCHNORR

Claus Schnorr's authentication and signature scheme [1396,1397] gets its security from the difficulty of calculating discrete logarithms. To generate a key pair, first

choose two primes, p and q, such that q is a prime factor of $p - 1$. Then, choose an a not equal to 1, such that $a^q \equiv 1 \pmod{p}$. All these numbers can be common to a group of users and can be freely published.

To generate a particular public-key/private-key key pair, choose a random number less than q. This is the private key, s. Then calculate $v = a^{-s} \bmod p$. This is the public key.

Authentication Protocol

(1) Peggy picks a random number, r, less than q, and computes $x = a^r \bmod p$. This is the preprocessing stage and can be done long before Victor is present.

(2) Peggy sends x to Victor.

(3) Victor sends Peggy a random number, e, between 0 and $2^t - 1$. (I'll discuss t in a moment.)

(4) Peggy computes $y = (r + se) \bmod q$ and sends y to Victor.

(5) Victor verifies that $x = a^y v^e \bmod p$.

The security is based on the parameter t. The difficulty of breaking the algorithm is about 2^t. Schnorr recommended that p be about 512 bits, q be about 140 bits, and t be 72.

Digital Signature Protocol

Schnorr can also be used as a digital signature protocol on a message, M. The public-key/private-key key pair is the same, but we're now adding a one-way hash function, $H(M)$.

(1) Alice picks a random number, r, less than q, and computes $x = a^r \bmod p$. This computation is the preprocessing stage.

(2) Alice concatenates M and x, and hashes the result:

$$e = H(M,x)$$

(3) Alice computes $y = (r + se) \bmod q$. The signature is e and y; she sends these to Bob.

(4) Bob computes $x' = a^y v^e \bmod p$. He then confirms that the concatenation of M and x' hashes to e.

$$e = H(M,x')$$

If it does, he accepts the signature as valid.

In his paper, Schnorr cites these novel features of his algorithm:

> Most of the computation for signature generation can be completed in a preprocessing stage, independent of the message being signed. Hence, it can be done dur-

ing idle time and not affect the signature speed. An attack against this preprocessing stage is discussed in [475], but I don't think it's practical.

For the same level of security, the length of signatures is less for Schnorr than for RSA. For example, with a 140-bit q, signatures are only 212-bits long, less than half the length of RSA signatures. Schnorr's signatures are also much shorter than ElGamal signatures.

Of course, practical considerations may make even fewer bits suitable for a given scheme: For example, an identification scheme where the cheater must perform an on-line attack in only a few seconds, versus a signature scheme where the cheater can calculate for years off-line to come up with a forgery.

A modification of this algorithm, by Ernie Brickell and Kevin McCurley, enhances its security [265].

Patents

Schnorr is patented in the United States [1398] and in many other countries. In 1993, PKP acquired the worldwide rights to the patent (see Section 25.5). The U.S. patent expires on February 19, 2008.

21.4 Converting Identification Schemes to Signature Schemes

There is a standard method of converting an identification scheme into a signature scheme: Replace Victor with a one-way hash function. The message is not hashed before it is signed; instead the hashing is incorporated into the signing algorithm. In principle, this can be done with any identification scheme.