

CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 17
March 21, 2012

Formalizing Zero Knowledge

- Computational Knowledge

- Composing Zero-Knowledge Proofs

Full Feige-Fiat-Shamir Authentication Protocol

Non-interactive Interactive Proofs

- Feige-Fiat-Shamir Signatures

Other Kinds of Interactive Proofs

- Interactive proof of graph non-isomorphism

Formalizing Zero Knowledge

Computational Knowledge

What does Bob learn from Alice?

We have seen several examples of zero knowledge proofs but no careful definition of what it means to be “zero knowledge”.

The intuition that “Bob learns nothing from Alice” surely isn’t true.

After running the FFS protocol, for example, Bob learns the quadratic residue x that Alice computed in the first step.

He didn’t know x before, nor did he and Alice know any quadratic residues in common other than the public number v .

By zero knowledge, we want to capture the notion that Bob learns nothing that might be useful in turning an intractable computation into a tractable one.

A general client process for interacting with Alice

Consider an arbitrary algorithm for performing some computation, i.e., suppose Mallory is trying to compute some function $f(z)$.

We regard Mallory as a probabilistic Turing machine with input tape and output tape.

- ▶ z is placed on the input tape at the beginning.
- ▶ If Mallory halts, the contents of the output tape is the answer.
- ▶ Mallory can also play Bob's role in some zero-knowledge protocol, say FFS for definiteness.
- ▶ During the computation, Mallory can read the number x that Alice sends at the start of FFS.
- ▶ Later, he can send a bit b to Alice.
- ▶ Later still, he can read the response y from Alice.
- ▶ After that, he computes and produces the answer, $f(z)$.

A Mallory-simulator

A Mallory-simulator, whom we'll call Sam, is a program like Mallory except he is not on the internet and can't talk to Alice.

Alice's protocol is *zero knowledge* if for every Mallory, there is a Mallory-simulator Sam that computes the same random function $f(z)$ as Mallory.

In other words, whatever Mallory does with the help of Alice, Sam can do alone.

The logical connection with knowledge

If Mallory computes some function with Alice's help (such as writing a square root of v to the output tape), then Sam can also do that without Alice's help.

Under the assumption that taking square roots is hard, Sam couldn't do that; hence Mallory also couldn't do that, even after talking with Alice.

We conclude that Alice doesn't release information that would help Mallory to compute her secret; hence her secret is secure.

Constructing a simulator

To show a particular interactive protocol is zero knowledge, it is necessary to show how to construct Sam for an arbitrary program Mallory.

Here's a sketch of how to generate a triple (x, b, y) for the FFS protocol.

$b = 0$: Sam generates x and y the same way Alice does—by taking $x = r^2 \bmod n$ and $y = r \bmod n$.

$b = 1$: Sam chooses y at random and computes $x = y^2 v \bmod n$.

What he can't do (without knowing Alice's secret) is to generate both triples for the same value x .

A simulator for FFS

Here's the code for Sam:

1. Simulate Mallory until he requests a value from Alice.
2. Save Mallory's state as Q .
3. Choose a random value $\hat{b} \in \{0, 1\}$.
4. Generate a valid random triple (x, \hat{b}, y) .
5. Pretend that Alice sent x to Mallory.
6. Continue simulating Mallory until he is about to send a value b to Alice.
7. If $b \neq \hat{b}$, reset Mallory to state Q and return to step 3.
8. Otherwise, continue simulating Mallory until he requests another value from Alice. Pretend that Alice sent him y and continue.
9. Continue simulating Mallory until he halts.

Properties of the simulator

The probability that $b = \hat{b}$ in step 7 is $1/2$; hence, the expected number of times Sam executes lines 3–7 is only 2.

Sam outputs the same answers as Mallory with the same probability distribution. **Requires some work to show.**

Hence, the FFS protocol is zero knowledge.

Note that this proof depends on Sam's ability to generate triples of both kinds without knowing Alice's secret.

Composing Zero-Knowledge Proofs

Serial composition

One round of the simplified FFS protocol has probability 0.5 of error. That is, Mallory can fool Bob half the time.

This is unacceptably high for most applications.

Repeating the protocol t times reduces error probability to $1/2^t$.

Taking $t = 20$, for example, reduces the probability of error to less than one in a million.

The downside of such *serial repetition* is that it also requires t round trip messages between Alice and Bob (plus a final message from Alice to Bob).

Parallel composition of zero-knowledge proofs

One could run t executions of the protocol in parallel.

Let (x_i, b_i, y_i) be the messages exchanged during the i^{th} execution of the simplified FFS protocol, $1 \leq i \leq t$.

In a *parallel execution*,

- ▶ Alice sends (x_1, \dots, x_t) to Bob,
- ▶ Bob sends (b_1, \dots, b_t) to Alice,
- ▶ Alice sends (y_1, \dots, y_t) to Bob,
- ▶ Bob checks the t sets of values he has received and accepts only if all checks pass.

Simulation proof does not extend to parallel execution

A parallel execution is certainly attractive in practice, for it reduces the number of round-trip messages to only $1\frac{1}{2}$.

The downside is that **the resulting protocol may not be zero knowledge by our definition.**

Intuitively, the important difference is that Bob gets to know all of the x_i 's before choosing the b_i 's.

Problem extending the simulator to the parallel case

While it seems implausible that this would actually help a cheating Bob to compromise Alice secret, the simulation proof used to show that a protocol is zero knowledge no longer works.

To extend the simulator construction to the parallel composition:

- ▶ First Sam would have to guess $(\hat{b}_1, \dots, \hat{b}_t)$.
- ▶ He would construct the x_i 's and y_i 's as before.
- ▶ When Mallory's program reaches the point that Mallory generates the b_i 's, the chance is very high that Sam's initial guesses were wrong and he will be forced to start over again. Indeed, the probability that all t initial guesses are correct is only $1/2^t$.

Full Feige-Fiat-Shamir Authentication Protocol

Full FFS overview

The full Feige-Fiat-Shamir Authentication Protocol combines ideas of serial and parallel execution to get a protocol that exhibits some of the properties of both.

A *Blum prime* is a prime p such that $p \equiv 3 \pmod{4}$.

A *Blum integer* is a number $n = pq$, where p and q are Blum primes.

If p is a Blum prime, then $-1 \in \text{QNR}_p$, so $\left(\frac{-1}{p}\right) = -1$. This follows from the Euler criterion, since $\frac{p-1}{2}$ is odd, so

$$(-1)^{\frac{p-1}{2}} = \left(\frac{-1}{p}\right) = -1.$$

If n is a Blum integer, then $-1 \in \text{QNR}_n$ but $\left(\frac{-1}{n}\right) = 1$.

Square roots of Blum integers

Let $n = pq$ be a Blum integer and $a \in \mathbb{QR}_n$. Exactly one of a 's four square roots modulo n is a quadratic residue.

Consider \mathbf{Z}_p^* and \mathbf{Z}_q^* . $a \in \mathbb{QR}_p$ and $a \in \mathbb{QR}_q$.

Let $\{b, -b\} = \sqrt{a} \pmod{p}$ and apply the Euler Criterion to both. Since

$$(-1)^{(p-1)/2} = -1 \quad \text{and} \quad b^{(p-1)/2} \in \{\pm 1\},$$

then either $b^{(p-1)/2} = 1$ or $(-b)^{(p-1)/2} = 1$.

Hence, either $b \in \mathbb{QR}_p$ or $-b \in \mathbb{QR}_p$. Call that number b_p .

Similarly, one of the square roots of $a \pmod{q}$ is in \mathbb{QR}_q , say b_q .

Applying the Chinese Remainder Theorem, it follows that exactly one of a 's four square roots modulo n is a quadratic residue.

Full FFS key generation

Here's how Alice generates the public and private keys of the full FFS protocol.

- ▶ She chooses a Blum integer n .
- ▶ She chooses random numbers $s_1, \dots, s_k \in \mathbf{Z}_n^*$ and random bits $c_1, \dots, c_k \in \{0, 1\}$.
- ▶ She computes $v_i = (-1)^{c_i} s_i^{-2} \bmod n$, for $i = 1, \dots, k$.
- ▶ She makes (n, v_1, \dots, v_k) public and keeps (n, s_1, \dots, s_k) private.

Notice that every v_i is either a quadratic residue or the negation of a quadratic residue.

It is easily shown that all of the v_i have Jacobi symbol 1 modulo n .

One round of the full FFS authentication protocol.

A round of the protocol itself is shown below. The protocol is repeated for a total of t rounds.

Alice		Bob
1. Choose random $r \in \mathbf{Z}_n - \{0\}, c \in \{0, 1\}.$ $x = (-1)^c r^2 \bmod n$	\xrightarrow{x}	
2.	$\xleftarrow{b_1, \dots, b_k}$	Choose random $b_1, \dots, b_k \in \{0, 1\}.$
3. $y = rs_1^{b_1} \dots s_k^{b_k} \bmod n.$	\xrightarrow{y}	
4.		$z = y^2 v_1^{b_1} \dots v_k^{b_k} \bmod n.$ Check $z \equiv \pm x \pmod{n}$ and $z \neq 0.$

Correctness of full FFS authentication protocol

When both Alice and Bob are honest, Bob computes

$$z = r^2(s_1^{2b_1} \cdots s_k^{2b_k})(v_1^{b_1} \cdots v_k^{b_k}) \bmod n.$$

Since $v_i = (-1)^{c_i} s_i^{-2}$, it follows that $s_i^2 v_i = (-1)^{c_i}$. Hence,

$$\begin{aligned} z &\equiv r^2(s_1^2 v_1)^{b_1} \cdots (s_k^2 v_k)^{b_k} \\ &\equiv x(-1)^c (-1)^{c_1 b_1} \cdots (-1)^{c_k b_k} \equiv \pm x \pmod{n}. \end{aligned}$$

Moreover, since $x \neq 0$, then also $z \neq 0$. Hence, Bob's checks succeed.

The chance that a bad Alice can fool Bob is only $1/2^{kt}$. The authors recommend $k = 5$ and $t = 4$ for a failure probability of $1/2^{20}$.

Zero knowledge property

Theorem

The full FFS protocol is a zero knowledge proof of knowledge of the s_j for $k = O(\log \log n)$ and $t = O(\log n)$.

Proof.

See U. Fiege, A. Fiat, and A. Shamir, *Zero knowledge proofs of identity*, ACM Symp. on Theory of Computing, 1987, 210–217. □

Non-interactive Interactive Proofs

Eliminating interaction from interactive proofs

Going from serial composition to parallel composition reduces communication overhead but may sacrifice of zero knowledge.

Rather surprisingly, one can go a step further and eliminate the interaction from interactive proofs altogether.

The idea is that Alice will provide Bob with a **trace of a pretend execution** of an interactive proof of herself interacting with Bob.

Bob will check that the trace is a valid execution of the protocol.

Of course, that isn't enough to convince Bob that Alice isn't cheating, for how does he ensure that Alice simulates random query bits b_i for him, and how does he ensure that Alice chooses her x_i 's before knowing the b_i 's?

Keeping Alice from cheating

The solution is to make the b_i 's depend in an unpredictable way on the x_i 's.

We base the b_i 's on the value of a “random-looking” hash function H applied to the concatenation of the x_i 's.

A non-interacting version of FFS

Here's how it works in, say, the parallel composition of t copies of the simplified FFS protocol.

- ▶ The honest Alice chooses x_1, \dots, x_t according to the protocol.
- ▶ Next she chooses $b_1 \dots b_t$ to be the first t bits of $H(x_1 \cdots x_t)$.
- ▶ Finally, she computes y_1, \dots, y_t , again according to the protocol.
- ▶ She sends Bob a single message consisting of $x_1, \dots, x_t, y_1, \dots, y_t$.
- ▶ Bob computes $b_1 \dots b_t$ to be the first t bits of $H(x_1 \cdots x_t)$ and then performs each of the t checks of the FFS protocol, accepting Alice's proof only if all checks succeed.

Why can't Alice cheat?

A cheating Alice can choose y_i arbitrarily and then compute a valid x_i for a given b_i .

If she chooses the b_i 's first, the x_i 's she computes are unlikely to hash to a string that begins with $b_1 \dots b_t$.¹

¹This assumes that the hash function “looks like” a random function. We have already seen artificial examples of hash functions that do not have this property.

Why can't Alice cheat?

If some b_i does not agree with the corresponding bit of the hash function, she can either change b_i and try to find a new y_i that works with the given x_i , or she can change x_i to try to get the i^{th} bit of the hash value to change.

However, neither of these approaches works. The former may require knowledge of Alice's secret; the latter will cause the bits of the hash function to change “randomly”.

Brute force cheating

One way Alice can attempt to cheat is to use a brute-force attack.

For example, she could generate all of the x_i 's to be squares of the y_i with the hopes that the hash of the x_i 's will make all $b_i = 0$.

But that is likely to require 2^{t-1} attempts on average.

If t is chosen large enough (say $t = 80$), the number of trials Alice would have to do in order to have a significant probability of success is prohibitive.

Of course, these observations are not a proof that Alice can't cheat; only that the obvious strategies don't work.

Nevertheless, it is plausible that a cheating Alice not knowing Alice's secret, really wouldn't be able to find a valid such "non-interactive interactive proof".

Contrast with true interactive proofs

With a true zero-knowledge interactive proof, Bob does not learn anything about Alice's secret, nor can Bob impersonate Alice to Carol after Alice has authenticated herself to Bob.

On the other hand, if Alice sends Bob a valid non-interactive proof, then Bob can in turn send it on to Carol.

Even though Bob couldn't have produced it on his own, it is still valid.

So here we have the curious situation that Alice needs her secret in order to produce the non-interactive proof string π , and Bob can't learn Alice's secret from π , but now Bob can use π itself in an attempt to impersonate Alice to Carol.

Feige-Fiat-Shamir Signatures

Similarity between signature scheme and non-interactive IP

A signature scheme has a lot in common with the “non-interactive interactive” proofs.

In both cases, there is only a one-way communication from Alice to Bob.

- ▶ Alice signs a message and sends it to Bob.
- ▶ Bob verifies it without further interaction with Alice.
- ▶ If Bob hands the message to Carol, then Carol can also verify that it was signed by Alice.

Not surprisingly, the “non-interactive interactive proof” ideas can be used to turn the Feige-Fiat-Shamir authentication protocol into a signature scheme.

Signature scheme from non-interactive IP

We present a signature scheme based on a slightly simplified version of the full FFS authentication protocol in which all of the v_i 's in the public key are quadratic residues, and n is not required to be a Blum integer, only a product of two distinct odd primes.

The public verification key is (n, v_1, \dots, v_k) , and the private signing key is (n, s_1, \dots, s_k) , where $v_j = s_j^{-2} \bmod n$ ($1 \leq j \leq k$).

Signing algorithm

To sign a message m , Alice simulates t parallel rounds of FFS.

- ▶ She first chooses random $r_1, \dots, r_t \in \mathbf{Z}_n - \{0\}$ and computes

$$x_i = r_i^2 \bmod n \quad (1 \leq i \leq t).$$

- ▶ She computes $u = H(mx_1 \cdots x_t)$, where H is a suitable cryptographic hash function.
- ▶ She chooses $b_{1,1}, \dots, b_{t,k}$ according to the first tk bits of u :

$$b_{i,j} = u_{(i-1)*k+j} \quad (1 \leq i \leq t, 1 \leq j \leq k).$$

- ▶ Finally, she computes

$$y_i = rs_1^{b_{i,1}} \cdots s_k^{b_{i,k}} \bmod n \quad (1 \leq i \leq t).$$

The signature is

$$s = (b_{1,1}, \dots, b_{t,k}, y_1, \dots, y_t).$$

Verification algorithm

To verify the signed message (m, s) , Bob computes

$$z_i = y_i^2 v_1^{b_{i,1}} \cdots v_k^{b_{i,k}} \bmod n \quad (1 \leq i \leq t).$$

Bob checks that each $z_i \neq 0$ and that $b_{1,1}, \dots, b_{t,k}$ are equal to the first tk bits of $H(mz_1 \cdots z_t)$.

When both Alice and Bob are honest, it is easily verified that $z_i = x_i$ ($1 \leq i \leq t$). In that case, Bob's checks all succeed since $x_i \neq 0$ and $H(mz_1 \cdots z_t) = H(mx_1 \cdots x_t)$.

Forgery

To forge Alice's signature, an impostor must find $b_{i,j}$'s and y_i 's that satisfy the equation

$$b_{1,1} \dots b_{t,k} \preceq H(m(y_1^2 v_1^{b_{1,1}} \dots v_k^{b_{1,k}} \bmod n) \dots (y_t^2 v_1^{b_{t,1}} \dots v_k^{b_{t,k}} \bmod n)).$$

where “ \preceq ” means string prefix. It is not obvious how to solve such an equation without knowing a square root of each of the v_i^{-1} 's and following essentially Alice's procedure.

Other Kinds of Interactive Proofs

Other kinds of interactive proofs

Not all interactive proofs follow this simple (x, b, y) pattern.

Suppose Alice wants to prove to Bob that G_0 and G_1 are *non*-isomorphic graphs.

Even ignoring questions of Alice's privacy, there is no obvious data that she can send Bob that will allow him to easily verify that the two graphs are not isomorphic.

However, under a different set of assumptions, Alice can convince Bob that they can't be isomorphic, even though Bob can't do so by himself.

An all-powerful teacher

In this version of interactive proof, we assume that Alice is all-powerful and can compute intractable problems. In particular, given two graphs, she can determine whether or not they are isomorphic.

Bob on the other hand has no extraordinary powers and can just perform computation in the usual way.

Alice uses her computational powers to distinguish isomorphic copies of G_0 from isomorphic copies of G_1 . If $G_0 \cong G_1$, there is no way she could do this, since any graph H isomorphic to one of them is also isomorphic to the other.

So by convincing Bob that she is able to reliably distinguish such graphs, she also convinces him that $G_0 \not\cong G_1$.

Interactive proof of graph non-isomorphism

Alice	Bob
1.	Choose random $b \in \{0, 1\}$. Compute a random isomorphic copy H of G_b .
2. If $H \cong G_0$ let $b' = 0$. If $H \cong G_1$ let $b' = 1$.	Check $b' = b$.

 \xleftarrow{H} $\xrightarrow{b'}$

Graph non-isomorphism IP is not zero-knowledge

Alice performs a computation for Bob that he could not do himself.

Namely, Alice willingly tells Bob for any H of his choosing whether it is isomorphic to G_0 or to G_1 .

(In any implementation of the protocol, she also probably tells him if H is not isomorphic to either one, perhaps by failing in step 2 when b' is undefined.)

Bit commitment

This protocol is an example of *bit-commitment*, another important cryptographic primitive that we will study later.

A *bit-commitment* is an encryption of a bit with a special property.

- ▶ The bit is hidden from anyone not knowing the secret key.
- ▶ There is only one valid way of decrypting it, no matter what key is used.

Thus, if $c = E_k(b)$:

- ▶ It is hard to find b from c without knowing k .
- ▶ For every k' , b' , if $E_{k'}(b') = c$, then $b = b'$.

Bit commitment intuition

In other words, if Bob produces a commitment c to a bit b , then b cannot be recovered from c without knowing Bob's secret encoding key k .

Also, there is no key k' that Bob might release that would make it appear that c is a commitment of the bit $1 - b$.

Non-isomorphism protocol viewed as bit commitment

In the non-isomorphism IP, H is a commitment of Bob's bit b .

Suppose Bob gives H to Carol (who doesn't have Alice's extraordinary computational powers).

Later Bob could convince Carol of his bit by telling her the isomorphism that proves $H \cong G_b$.

But there is nothing he could do to make her believe that his bit was really $1 - b$ since $H \not\cong G_{1-b}$.

The actual protocol doesn't use the commitment in quite this way. Rather than having Bob later reveal his bit, Alice uses her special powers to discover the bit committed by H .