CSCI 1515: Applied Cryptography

P. Miao

Spring 2023

These are lecture notes for CSCI 1515: Applied Cryptography taught at Brown University by Peihan Miao in the Spring of 2023.

These notes are taken by Jiahua Chen with gracious help and input from classmates and fellow TAs. Please direct any mistakes/errata to me via email, post a thread on Ed, or feel free to pull request or submit an issue to the notes repository (https://github.com/BrownAppliedCryptography/notes).

FYI, todo's are marked like this.

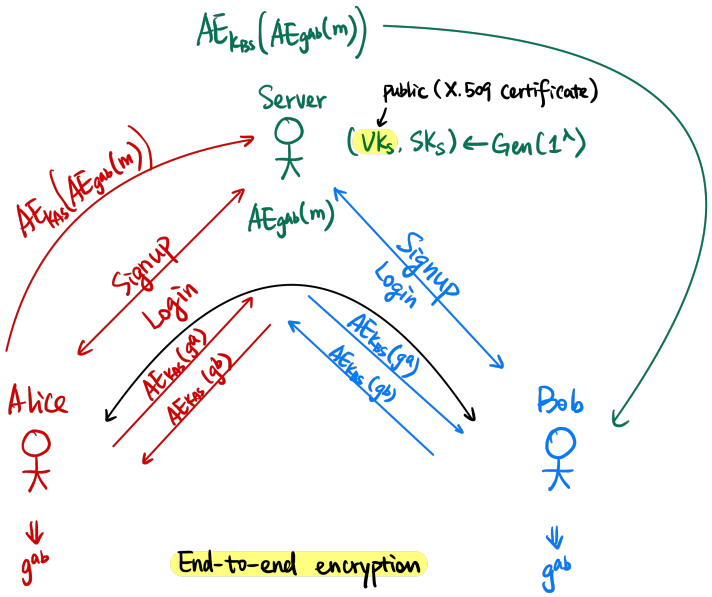
Notes last updated March 7, 2023.

Contents

§1 February 28, 2023

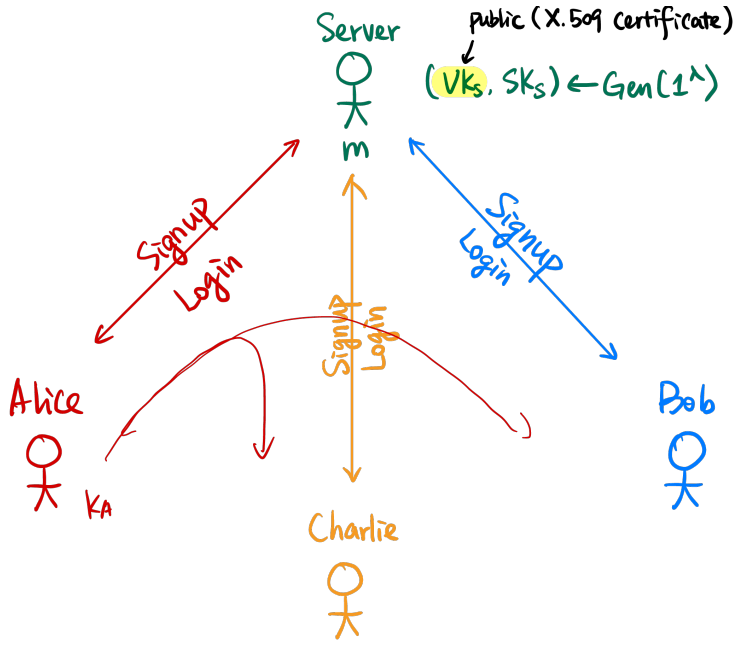
§1.1 Secure Messaging, *continued*

This was just a review of **??**. The server with a known verification key will sign public keys for Alice and Bob. Alice and Bob will exchange keys via the server and communicate messages via the server, encrypted with their shared secret. This allows the server to pass messages that remain secret to the server.



§1.1.1 Group Messaging

When we move to group chats, there are more things we need to consider. For example, do we want to reveal this message to the server? In this case, Alice can send the message in the clear to the server and it is forwarded. Additionally, we might ask whether we want to hide the group structure from the server.



In general, there are two paradigms for group messaging. Either everyone uses the *same* key, or everyone has a different key. In WhatsApp, Alice would use a symmetric ratchet with key A, gr (Alice's key and group key) to send the message to the server, and WhatsApp will forward the same encrypted message to Bob and Charlie. While the group structure is revealed to the server, but the message contents are unbeknownst to the server.

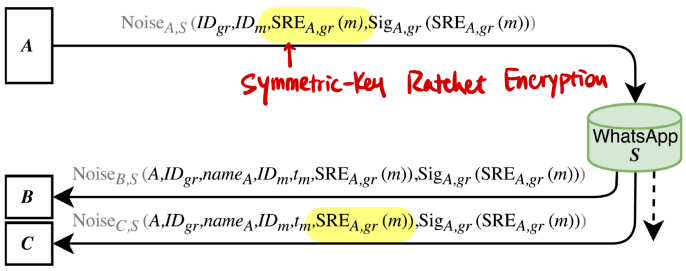


Figure 5. Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in WhatsApp.

In Signal, on the other hand, every pair of users has a differnt key. If Alice wants to send a message to Bob and Charlie, Alice will encrypt two messages, one with Alice/Bob's key and another with Alice/Charlie's key. The server will forward the encrypted messages to the users respectively. In Signal, a double ratchet encryption is preformed between every pair of parties. Another guarantee is that the group structure can be hidden against the server—Alice sending individual messages to Bob and Charlie is indistinguishable from their group texts.

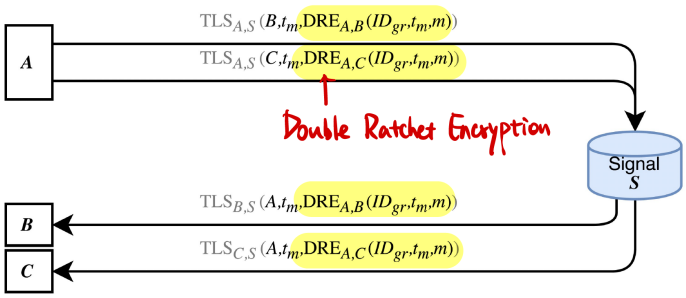
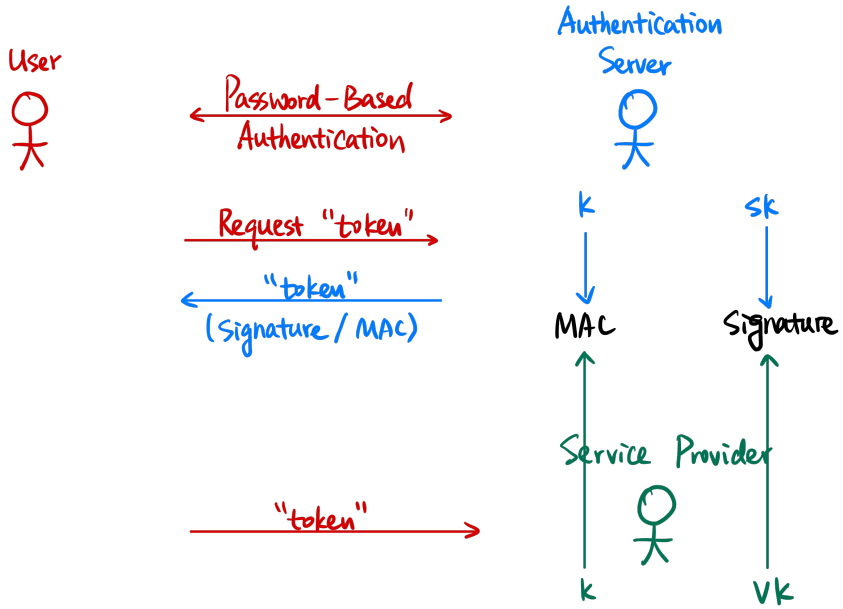


Figure 3. Schematic depiction of Signal's traffic, generated for a message m from sender A to receivers B and C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$. Transport layer protection is not in the analysis scope (gray).

§1.2 Single Sign-On (SSO) Authentication

Often, we'll 'log in with Google' or 'log in with Apple'[1]. A user will authenticate themselves with the authentication server (Google, Apple, Shibboleth), and will be issued a 'token' (usually a signature/MAC) for them to then authenticate themselves against the service provider.

Implementations include OAuth or OpenID, which is the format used by Google/Apple/Facebook, etc. Within enterprises, Kerberos credentials allow for SSO as well as things such as printing, connecting to servers, etc.



[1] Even Brown has Shibboleth!

§1.3 Zero-Knowledge Proofs

As mentioned in our course outline, a Zero-Knowledge Proof (ZKP) is a scheme that allows a prover to prove to a verifier some knowledge that they have, without revealing that knowledge.

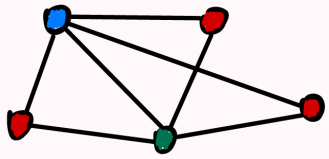
What is a proof? We consider what a 'proof system' is. For example, we'll have a *statement* and a *proof* that is a purported proof of that statement. What guarantees do we want from this proof system? If the statement is true, we should be able to prove it; and if the statement is false, we shouldn't be able to prove this. These are our guarantees of *completeness* and *soundness*.

Completeness. If a statement is true, there exists a proof that proves it is true.

Soundness. If a statement is false, any proof cannot prove it is true.

We can think of NP languages from a proof system perspective.

Example 1.1 (Graph 3-Coloring)
Consider the *Graph 3-coloring*.



We define our language

$$L = \{G : G \text{ has a 3-coloring}\}$$

and relation

$$R_L = \{(G, 3\text{Col})\}$$

Our statement will be that G has a 3-coloring. Our proof is providing such a coloring $(G, 3\text{Col}) \in R_L$.

This satisfies completeness and soundness. Every 3-colorable graph has a proof that is the 3-coloring itself, and if a graph doesn't have a 3-coloring, it will not have a proof.

We can think of NP languages as a proof system. A language L is in NP if $\exists$poly-time$V$ (verifier) such that

Completeness. $\forall x \in L, \exists w$ (witness) such that $V(x, w) = 1$.

Soundness. $\forall x \notin L, \forall w^*, V(x, w^*) = 0$.

The prover will prove to the verifier that they have knowledge of witness w without revealing the witness itself.

Definition 1.2 (Zero-Knowledge Proof System)

Let (P, V) (for *prover* and *verifier*) be a pair of probabilistic poly-time (PPT) interactive machines. (P, V) is a zero-knowledge proof system for a language L with associated relation R_L if

Completeness. $\forall (x, w) \in R_L$, $\Pr[P(x, w) \leftrightarrow V(x) \text{ outputs } 1] = 1$. That is, if there is a $x \in L$ with witness w, a prover will be able to prove to the verifier that they have knowledge of w.

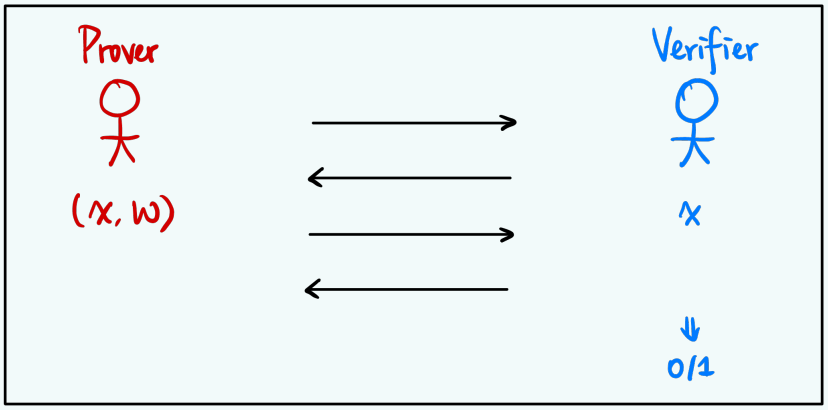
Soundness. $\forall x \notin L$, $\forall P^*$, $\Pr[P^*(x) \leftrightarrow V(x) \text{ outputs } 1] \simeq 0$. That is, for every x not in the language, our prover P^* will not be able to prove its validity to V, with negligible probability. If P^* is PPT, we call the system a *zero-knowledge argument*.

We need an additional property that this is actually *zero-knowledge*[2]. We want to say that the verifier is unable to extract any additional information from the interaction between the verifier and prover. That is, even without the witness, a verifier might be able to 'simulate' this transaction *by themselves*!

We'll say $\forall$PPT $V^*, \exists$PPT S such that $\forall (x, w) \in R_L$,

$$\text{Output}_{V^*}[P(x, w) \leftrightarrow V^*(x)] \simeq S(x).$$

That is to say, for everything in the language, the output transcript between the prover and verifier can be *simulated* by the simulator without knowledge of the witness[3].



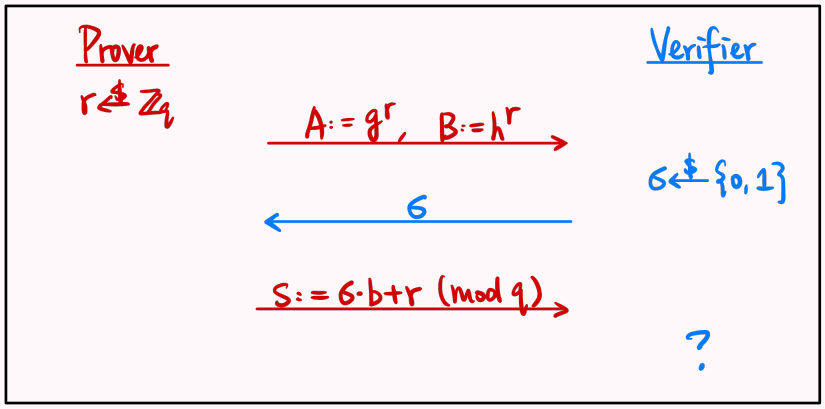
[2]That is, the prover could just send the witness in the clear to the verifier, which satisfies completeness and soundness.

[3]This is *counterintuitive*, because if any PPT can simulate the proof by themselves, how do we know we're even talking to a prover that has a witness? This is subtle, but we give extra power to the simulator that they are

Example 1.3 (Diffie-Hellman Tuple)

We want to prove that $h = g^a, u = g^b, v = g^{ab}$ is a Diffie-Hellman Tuple in a cyclic group $\mathbb{G}$ of order q and generator g.

Our witness is 'private exponent' b. Our statement is that $\exists b \in \mathbb{Z}_q$ such that $u = g^b$ and $v = h^b$.



The prover will randomly sample $r \xleftarrow{\$} \mathbb{Z}_q$ and send to the verifier $A := g^r$ and $B := h^r$. The verifier randomly samples *challenge* $\sigma \xleftarrow{\$} \{0, 1\}$, and sends this challenge bit to the prover. The prover will respond with $s := \sigma \cdot b + r \pmod q$. If the challenge bit was 0, $s = r$ and the verifier verifies $A = g^S$ and $B = h^S$. If the challenge bit was 1, $s = b + r$ and the verifier verifies $u \cdot A = g^s$ and $v \cdot B = h^S$.

Completeness: If this statement is true, the prover will be able to convince the verifier since they have knowledge of b.

Soundness: If the statement is *not true*, what is the probability that the prover will be able to convince the verifier? When $\sigma = 0$, then it's easy for the prover to pass validation. When $\sigma = 1$, is it possible for the prover to send back a valid s (or, knowing that $\sigma = 1$, can they generate a first round message that makes the message valid in the third round). For example, the prover will first randomly sample s and compute $A = g^s \cdot u^{-1}$ and $B = g^s \cdot v^{-1}$ to send in the first round. However, the prover cannot *simultaneously* pass $\sigma = 0$ *and* $\sigma = 1$ without knowing b. So the probability that the prover can convince the verifier is exactly $\frac{1}{2}$.

Can we bootstrap this to be negligible? We repeat this protocol λ times and the probability will be $\frac{1}{2^\lambda}$. r, σ are randomly sampled each time. Note that we are running the protocol a polynomial number of times for an exponential decrease in probability.

We can deconstruct this probability of $\frac{1}{2}$. If the prover sends $A = g^r, B = h^r$, if $\sigma = 0$, the prover can prove this. If $\sigma = 1$, the prover will need to produce such s such that $u \cdot g^r = g^s$ and $v \cdot h^r = h^s$,

allowed to *rewind* the verifier to some previous step. If the transcript can be simulated, then surely no information is leaked from the protocol.

but then this reduces to solving for b (we can get $b = s - r$ out of such an s). Otherwise, if $A = g^{r_1}$ and $B = g^{r_2}$, and $\sigma = 0$, we'll 'catch' the phony prover.

In essence, the prover is using a one-time pad r to mask b, but when prompted, the prover can also be asked to reveal r in the clear that it is valid.

What might a simulator do? Since the simulator can 'rewind' the verifier to a previous step. The simulator will guess whether the challenge bit, and prepare the first round message. If the guess is correct, we proceed. If not, the simulator will just rewind and do it again until it is correct. This gives the basis of our intuition behind the zero-knowledge aspect of this proof system. Rewinding λ times per iteration (at most) gives us $1 - \frac{1}{\lambda^2}$ times that it will succeed, and we repeat this over λ iterations.

To bootstrap the soundness error, what if we did the entire procedure in parallel? So there will be a total of 3 messages, each containing λ iterations of information. In particular, is it still zero-knowledge? We can't rewind anymore, since we need to get λ choice bits correct, which means we rewind an exponential number of times (violating PPT simulator). *Maybe there are other ways to construct simulators...* There are some questions:

1. Can we prove that this is zero-knowledge?

2. If not, can we disprove that this is not zero-knowledge?

3. Can we, as the verifier, get more information from this protocol running in parallel?

This has been an open problem for many years—it was recently proved that this is *not* zero-knowledge. However, it is still not clear whether we can extract more information from this protocol running in parallel.

§2 March 2, 2023

§2.1 Zero Knowledge Proofs, *continued*

§2.1.1 Recap

Recall that a zero-knowledge proof is an interactive proof system between two parties: a *prover* and a *verifier*. We say that (P, V) a pair of probablistic poly-time (PPT) interactive machines is a *zero-knowledge proof system* for a language L with associated relation R_L if we have

Completeness. If $\forall(x, w) \in R_L$ (x and witness w), the prover will always be able to prove that this is true.

Soundness. If $\forall x \notin L$, the prover cannot convince the verifier that $x \in L$.

Furthermore, we have a zero-knowledge definition, which is a bit counterintuitive. To guarantee that the verifier doesn't learn anything from the system, it means that a simulator that doesn't know the witness w can simulate the entire transcript (all transactions between the prover and verifier). That it, since a simulator can simulate this, the verifier had better not learn any additional information.

Example 2.1

We saw a quick example of a zero-knowledge proof for the Diffie-Hellman tuple. Our witness b is the Diffie-Hellman exponent, and inputs are g^a, g^b, g^{ab}.

The prover will generate a mask $r \xleftarrow{\$} \mathbb{Z}_q$. The prover sends $A := g^r, B := h^r$ to the verifier. The verifier issues a challenge bit, and the prover either provides the mask $s := r$ if $\sigma = 0$, otherwise the prover will provide b masked with r $s := b + r$.

We'll detail some of the desired properties of this zero-knowledge proof:

Completeness. Completeness is straightforward, since the prover will always convince the verifier.

Formally, $\forall(x, w) \in R_L$ (any x with witness w in language), $\Pr[P(x, w) \leftrightarrow V(x) \text{ outputs } 1] = 1$.

Soundness. If the statement is not true, the prover cannot convince to the verifier. The prover can only convince the verifier with probability $\frac{1}{2}$ on every iteration, repeating λ iterations makes this probability $\frac{1}{2^\lambda}$ (we want this! we want negligible probability that the prover succeeds).

Formally, $\forall x \notin L$, $\forall$PPT P^* (a malicious prover, who is trying to prove $x \in L$), $\Pr[P^*(x) \leftrightarrow V(x) \text{ outputs } 1] \simeq 0$.

Zero Knowledge. Since the simulator can 'rewind time', it will guess the challenge bit σ. If it is correct, it'll proceed, otherwise it will rewind until the correct σ is chosen. This takes at worst[4] $O(\lambda^2)$ attempts over λ.

Formally, $\forall$PPT V^* (for any verifier, even one acting maliciously), $\exists$PPT S (exists a simulator) such that $\forall(x, w) \in R_L$, the simulator's output is computationally indistinguishable from the output $\text{Output}_{V^*}[P(x, w) \leftrightarrow V^*(x)]$ (whatever the verifier outputs in the real world).

§2.1.2 Proof of Knowledge

We missed something very subtle in our soundness guarantee. Consider a case where *every* $x \in L$. Our soundness guarantee is moot here—the prover will never be attempting to prove something false. Yet, our model doesn't require the prover to actually know the witness w.

There is an even stronger assumption we can make, *Proof of Knowledge* that describes protocols where the prover needs to actively know a witness, not just that a certain element is in the language.

We define Proof of Knowledge similarly to Zero Knowledge property.

An extractor, interacting with a prover (not necessarily honest), should be able to *extract* the witness w out of its communication with the prover, with the additional power that it can rewind the prover.

Example 2.2

How might an extractor get witness b in the Diffie-Hellman example?

The extractor can first pick $\sigma = 0$, which gives them s such that $A = g^s, B = h^s$. Then, the extractor rewinds the protocol and issues challenge $\sigma' = 1$, gaining s' such that $u \cdot A = g^{s'}$ and $v \cdot B = h^{s'}$.

Then, $u = g^{s-s'}$ and $v = h^{s-s'}$, combining these they can extract valid $b = s - s' \pmod q$. If the prover can always convince the verifier, then the extractor will always be able to extract the witness w.

Formally, $\exists$PPT E (called *extractor*) such that $\forall P^*$ (potentially dishonest prover), $\forall x$,

$$\Pr[E^{P^*(\cdot)}(x) \text{ outputs } w \text{ s.t. } (x, w) \in R_L] \simeq \Pr[P^* \leftrightarrow V(x) \text{ outputs } 1].$$

This is to say, the probability that the extractor can extract a witness is computationally indistinguishable from the probability of the prover successfully proving $x \in R_L$.

[4] That is, the probability of reaching a $O(\lambda^2)$ runtime is negligible in λ.

So we've built up our four properties:

- Completeness: The prover can prove whenever $x \in R_L$.
- Soundness: For any x not in R_L, the prover can only prove $x \in R_L$ with *negligible* probability.
- Zero Knowledge: The verifier does not gain any additional information from the proof. That is, a simulator could have 'thought up' the entire transcript in their head given the ability to rewind.
- Proof of Knowledge: An even stronger guarantee than soundness (this implies soundness)—a prover must have the witness in hand to be able to prove $x \in R_L$. That is, an extractor could interact with the prover (and rewind) to be able to extract the information of w from the interaction.

§2.1.3 Schnorr's Identification Protocol

Example 2.3 (Schnorr's Identification Protocol)

We saw a variant earlier with the Diffie-Hellman triple proof. This is the general form.

The input is a cyclic group $\mathbb{G}$ of order q, with generator g and $h = g^a$.

The prover first samples a mask $r \xleftarrow{\$} \mathbb{Z}_q$, and sends $A : g^r$ to the verifier. The verifier issues challenge $\sigma \xleftarrow{\$} \mathbb{Z}_q$[5]. The prover will reply with $s := \sigma \cdot a + r \pmod q$.

The verifier will verify that $g^s \stackrel{?}{=} h^\sigma \cdot A$ and $g^s = g^{\sigma \cdot h + r} \stackrel{?}{=} (g^a)^r \cdot g^r$.

We show the desired properties:

Completeness. Complete

§2.2 Sigma Protocols

Complete

[5]Note that before we only had $\sigma \in \{0, 1\}$, but now σ is in the larger space

§3 March 7, 2023