An identification scheme is a mechanism for Alice to prove her identity to Bob. It consists of a setup phase in which some public or private information is shared between Alice and Bob and an identification phase in which Alice's identity is established. The setup phase occurs, for example, when you register a password to a new website, or when you are issued a uOttawa smart card. Identification occurs when you log in with your password, or when you swipe your smart card at a reader.

1 Secret-key identification

In a secret-key identification scheme the setup phase consists of a common random choice of a secret key k shared between Alice and Bob. It is common to call Alice *the Prover* and Bob *the Verifier*: Alice's job is to prove her identity to Bob. Bob must verify Alice's claimed identity.

Example 1: Password authentication. In the setup phase, Alice registers her password k with Bob. In the identification phase, Alice sends k to Bob. Bob accepts if the password matches the record.

Plain password authentication is insecure against an Eve who can observe the interaction. Once Eve gets hold of Alice's password she can pass verification.

This type of *passive attack* can be modeled by a two-phase game (see Figure 1 (a)). In the learning stage, Eve observes some number of authentication attempts by Alice, as well as Bob's responses. In the validation stage, Eve takes up the role of a *cheating prover* P^*. She succeeds if Bob accepts her authentication attempt.

In plain password authentication, Eve intercepts Alice's password k in the learning stage, and then *replays* it to pass validation. Every (stateless) deterministic identification scheme is subject to this passive attack.

Example 2: Alice and Bob use the shared key k to index a PRF F. In the identification phase, Alice sends a random x together with the PRF value $F(x)$. Bob validates the identification attempt (x, y) if y equals $F(x)$.

This scheme is still vulnerable to a replay attack. Suppose Eve eavesdrops the pair $(x, F(x)) = (\texttt{010110}, \texttt{110010})$. Eve can later pass verification with the very same pair. (Stateless) Bob cannot distinguish between an input x that was freshly sampled by Alice from one that was replayed by Eve.

Any randomized identification scheme in which the randomness is controlled by the prover is vulnerable to a replay attack for this reason. The verifier cannot tell whether the randomness is authentic or replayed. Security against passive attacks requires that the randomness be controlled by the Verifier.

Secret key challenge-response:

1. Verifier sends a random challenge x to Prover.
2. Prover responds with $y = F(x)$.
3. Verifier accepts if y equals $F(x)$.

To pass identification, Eve now has to submit the PRF value $F(x^*)$ for a challenge x^* chosen by Bob. Provided that the PRF domain is sufficiently large, Eve is unlikely to have eavesdropped on this challenge before. As F is pseudorandom, the eavesdropped challenge-response pairs $(x, F(x))$ are of no use to her in guessing $F(x^*)$.

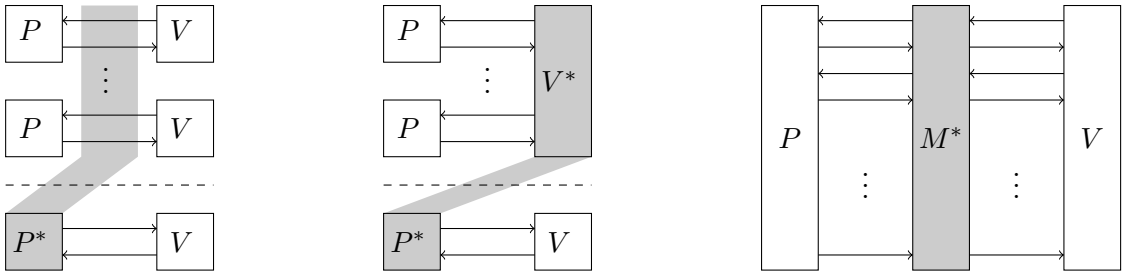

FIGURE 1: Identification attacks: (a) passive; (b) active; (c) woman-in-the-middle. The learning and validation stages are separated by the dashed line.

There is a more powerful type of attack in which Eve impersonates the *verifier* (see Figure 1 (b)). In an active attack Eve not only observes the messages sent by Alice, but she controls those produced by Bob. Phishing scams in which an inauthentic website tries to steal users' credentials fit into this model.

The challenge-response protocol is secure even against active attacks. Even if Eve gets to pick Bob's challenges in the learning stage, all she ends up learning are values of the PRF F at inputs of her choice. Unless the validation challenge x^* matches one of them, she has effectively no better strategy than guessing $F(x^*)$ at random.

The most powerful adversary is a *woman in the middle* who can alternate between playing prover and verifier roles throughout the execution (see Figure 1 (c)). Identification that is secure against woman-in-the-middle attacks is impossible unless additional assumptions are made about the underlying communication model.

2 Public-key identification

In public-key identification, during the setup phase Alice generates a key pair: a secret key sk that she gets to keep and a public key pk that is shared with Bob and Eve. The functionality requirement is that after interacting with Alice, Bob accepts with probability one. Since the only distinguishing feature of Alice is her public key, a public-key identification scheme is a *proof of knowledge*: Alice has to convince Bob that she knows her corresponding secret key.

As in the private-key setting, there are passive and active attacks. The difference here is that Eve also knows the public key. To thwart these attacks Alice and Bob must at the very least ensure that Eve does not learn it.

What happens when we try to adapt the challenge-response protocol to the public key setting? There is no real public-key analogue of a pseudorandom function. Its closest relatives are public-key encryption and digital signatures. Let's start with public-key encryption.

Encrypted challenge-response:

1. Verifier chooses a random message x and sends the encryption $c = Enc_{pk}(x)$.
2. Upon receiving c, Prover decrypts and responds with $x' = Dec_{sk}(c)$.
3. Verifier accepts if $x' = x$.

Encrypted challenge-response is functional as long as the underlying encryption scheme is. It is also secure against passive attacks, provided that the encryption is message indistinguishable. All Eve gets to observe in a session is the encryption $Enc_{pk}(x)$ of a random message x, followed by x itself. This is a "random variable" that she could have sampled on her own, provided that she started with x and encrypted it under Alice's public key. The eavesdropped data give her no advantage in decrypting a fresh challenge c^*.

Impersonation attacks are trickier to reason about. Suppose that the encryption is ElGamal. Alice's key pair is $sk = A, pk = g^A$. Bob encrypts his challenge x as $(g^Y, x \cdot pk^Y)$ and asks Alice to decrypt it. Now suppose Eve impersonates Bob. She can then effectively ask Alice to decrypt any ciphertext of her choosing. Eve could, for instance, ask for a decryption of $(pk^{-1}, 1) = (g^{-A}, 1)$. Alice would decrypt this to $1 \cdot (g^{-A})^{-A} = g^{A^2}$. Eve has effectively managed to exponentiate the *square* of Alice's secret key. This is not something that she could have easily calculated on her own.

In conclusion, if Eve can get Alice to decrypt messages of her choice, she can learn unintended information about Alice's secret key. In the case of ElGamal encryption, it is unclear if this information can help her later impersonate Alice. There are related examples in which challenge-response is insecure against active attacks even though the underlying encryption scheme is message indistinguishable.

Identification from signatures

Digital signatures and identification share the goal of establishing authenticity. The difference is that digital signatures are meant to authenticate data, while identification schemes are for establishing identity. Nevertheless, they both establish knowledge of the signer's/identifier's secret key while preventing the attacker from learning anything useful about it.

To repurpose a signature scheme for identification, the verifier asks the prover to sign a random challenge.

Signed challenge-response:

1. Verifier sends a random message x.
2. Prover responds with $s = Sign_{sk}(x)$.
3. Verifier runs $Ver_{pk}(x, s)$.

Signed challenge-response is secure against passive attacks. All Eve gets to observe in the learning stage are random messages and their signatures. They do not help her forge the signature to a fresh challenge in the validation stage. An active attack gives Eve the additional ability to observe signatures of messages of her choice. Unless the validation challenge x^* happens to match one of them by chance, she gains no additional advantage in forging a signature to x^*.

3 Schnorr's identification scheme

Schnorr came up with an ingenious twist on ElGamal-based challenge-response. As usual, g is a quadratic residue modulo a (large) safe prime p, and all exponents are taken modulo $q = (p-1)/2$. Alice's secret key is A and her public key is $pk = g^A$. All base arithmetic is modulo p.

Schnorr's identification scheme:

1. Prover chooses a random B and sends the *commitment* $c = g^B$ to Verifier.
2. Verifier sends a random challenge X to Prover.
3. Prover responds with $Y = A \cdot X + B \bmod q$.
4. Verifier accepts if $g^Y \equiv pk^X \cdot c$.

Schnorr's identification scheme is functional: g^Y equals $g^{A \cdot X + B} = (g^A)^X \cdot g^B = pk^X \cdot c$.

In an passive attack, Eve observes the commitment g^B, the challenge X, and the response $Y = A \cdot X + B$ exchanged during an identification session, all relative to Alice's public key $pk = g^A$. Eve's "view" for a given public key g^A is summarized by the random variable $(g^B, X, Y = A \cdot X + B)$.

Eve could have sampled the same random variable on her own in a different order. Since X and B are random and independent, so are X and Y. If Eve samples X and Y first, then she can deduce g^B as $g^{Y - A \cdot X} = g^Y \cdot pk^{-X}$. What she observed from the interaction is something that she could have sampled by herself. Eve learned nothing by observing a successful identification attempt.

In the validation stage, Eve plays the role of the prover. She gets to choose her commitment c. It no longer has to be random. Her response Y to Bob's challenge X can also be chosen in any way she likes. Suppose that she has concocted some devious strategy that passes validation, namely she has come up with X and Y such that

$$g^Y \equiv c \cdot pk^X.$$

If she managed to pass validation once, she can pass a second time. This time, the challenge X' is likely to be different (and so was probably her response):

$$g^{Y'} \equiv c \cdot pk^{X'} \qquad \text{for some } X' \neq X.$$

Eve has found X, X', Y, and Y' so that

$$\frac{g^{Y'}}{g^Y} \equiv \frac{c \cdot pk^{X'}}{c \cdot pk^X} \equiv \frac{pk^{X'}}{pk^X},$$

or $g^{Y'-Y} \equiv pk^{X'-X}$, or

$$g^{(Y'-Y)(X'-X)^{-1}} \equiv pk \tag{1}$$

Eve has calculated the discrete log of Alice's public key. But she doesn't have the resources to do that. Eve has no better strategy to impersonate Alice than cracking her secret key.

Impersonation attacks are again trickier to reason about. Eve can, for example, set her challenge to be the positive square root $\sqrt{c} = \sqrt{g^B}$ of the commitment. She then learns $Y = (A\sqrt{g^B} + B) \bmod q$. When B is random, this is a strange-looking random variable that might reveal information about the secret key A. While no useful active attacks of this type are known, it is not known if Schnorr's scheme is immune to them all. In conclusion, Schnorr's identification scheme is plausibly secure against active attacks, but its security requires stronger assumptions than the intractability of discrete logarithms.

4 The Fiat-Shamir heuristic

Bob's role in Schnorr's identification scheme is a little silly. All he is asked to do is to pick a random challenge (and then verify it at the end). It is tempting for him to outsource this action to Alice. The benefit is that the scheme becomes *noninteractive*. Alice sends all information—commitment, challenge, and response—in one shot.

Earlier we saw that allowing Alice to choose her own challenge would cede too much control to Eve. Eve can replay Alice's identification session and pass verification.

Amos Fiat and Adi Shamir (based on an idea of Manuel Blum) proposed a compromise: The challenge should be picked by Alice, but not at random. The challenge X should be set deterministically to $h(c)$, where h is some unstructured hash function like SHA-3.

The resulting scheme is shown in Figure 2 (b). Alice picks a random B, commits to $c = g^B$, sends the *hash value* $X = h(c)$, and follows up with $Y = A \cdot X + B$. Bob computes $X = h(c)$ at his end and verifies that $g^Y \equiv pk^X \cdot c$ as in Schnorr's identification scheme.

This scheme is no longer secure because it is deterministic. Eve can replay an eavesdropped interaction. But this is essentially the only attack that is available to her! If she tries to come up with a commitment c^* that is different from the ones that she eavesdropped on, she is out of luck. As the hash h is unstructured,

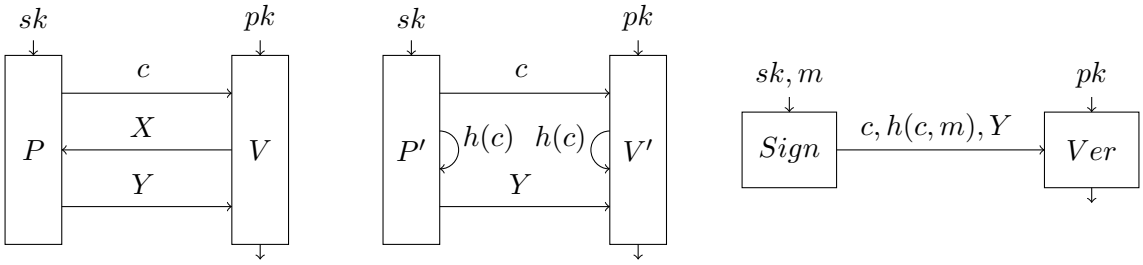

FIGURE 2: Digital signatures from identification: (a) Schnorr's identification scheme; (b) the Fiat-Shamir heuristic; (c) Schnorr's signature scheme.

the challenge $h(c^*)$ will appear random and independent from all previous challenges, just like in Schnorr's identification scheme. We already argued that Eve cannot pass verification on random challenges short of finding the discrete log of Alice's public key.

This is very close to the security requirement of a digital signature. Replayed message-signature pairs do not qualify as forgeries. To turn the non-interactive variant of Schnorr's scheme into a digital signature, all that remains to do is to bind the message to its signature. The hash function already does this job (see Figure 2(c)).

Schnorr's signature scheme

Key generation: Alice's secret key is A. Her public key is $g^A \bmod p$.

Signing: To sign m, Alice picks a random B, calculates $c = g^B$, $X = h(c, m)$, and $Y = AX + B$ and sends out the signature (c, Y).

Verification: Bob accepts message-signature pair $(m; c, Y)$ if $g^Y = pk^{h(c,m)} \cdot c$.

The functionality of Schnorr signatures is inherited from the identification scheme. When Alice submits an authentic signature, both sides in Bob's verification equation evaluate to $g^{A \cdot h(c,m)+B}$.

Now suppose Eve intercepts the message-signature pair $m; g^B, Y = A \cdot h(g^B, m) + B$. Unless Alice happened to have picked the same B before, from her perspective, $X = h(g^B, m)$ will look to her like a fresh random value that is independent of B. It is as if she had eavesdropped on a session of Schnorr's identification scheme with challenge X. We already argued that eavesdropping on such a session does not teach her anything new. The only way for her to forge is to satisfy Bob's verification equation $g^Y = pk^{h(c,m)} \cdot c = g^{Ah(c,m)} \cdot c$ out of thin air. As long as she can do this in a way that does not depend on the specific instantiation of h, she can run her attack twice and find the discrete log of pk as in (1).

5 Zero-knowledge proofs of knowledge

An identification scheme is an example of a proof of knowledge. In such a proof the Prover wants to convince the Verifier that she knows some secret information such as the discrete log of her public key pk. One way to accomplish this is to send the secret key in the clear. This certainly convinces Bob that Alice knows her secret key, but Alice's key is no longer secret. Bob, for one, will learn it, even if Eve is absent from the interaction.

Schnorr's identification scheme is also a proof of knowledge of Alice's secret key A. We argued that anyone who can pass verification (twice on different challenges) can deduce A. The only way to pass reliably is to know A in the first place.

Schnorr's scheme, however, has an important advantage over merely revealing A. Even though, at the end of the interaction, Bob is convinced that Alice knows her secret key, he did not learn it himself. All Bob saw was his own random challenge X followed by Alice's independent random response $Y = AX + B$,

preceded by Alice's commitment $c = g^B$. From his perspective, X and Y look purely random, and c can be derived from them as $g^Y pk^{-X}$. What he observed are two random numbers preceded by some fixed function of them. He could have sampled all of this on his own. It reveals not one extra bit of information about Alice's secret key.

A proof of knowledge is said to be *zero-knowledge* if the verifier learns no *new* information about the secret. In general, the prover may have had some side information before the interaction, for instance whether A is even or odd. Zero-knowledge stipulates that this is all he knows after the interaction. Whatever random messages were exchanged look like a random variable that he could have *simulated* on his own.

Schnorr's scheme is a zero-knowledge proof of knowledge of the discrete logarithm $f(X) = g^X$. There is nothing special about discrete logarithms in this respect. For any function f that can be calculated with reasonable effort, Alice can prove knowledge of a secret input x that maps to $y = f(x)$ without revealing any information about it.

Zero-knowledge proofs of knowledge have many applications in cryptography. Here is one illustrative example. A company wants a system for filing complaints. The complaints must be authored by a legitimate employee. One way to ensure this is to ask that the complaint be signed by an employee's secret key. The signature would then reveal the employee's identity. Few employees would be brave enough to risk that.

It is therefore desirable that the system for filing complaints be anonymous. The company's requirement is that the complaint m be signed by *some* employee. The signature s need not reveal which one. The actual employee then needs to prove that she knows how to produce a signature that authenticates under one of the employees' public keys:

$$\text{I know } s \text{ so } Ver_{\texttt{Alice}.pk}(m, s) \text{ accepts or } Ver_{\texttt{Bob}.pk}(m, s) \text{ accepts } \cdots \text{ or } Ver_{\texttt{Zach}.pk}(m, s) \text{ accepts}$$

A zero-knowledge proof of knowledge gives an employee the ability to certify that she has created the requisite signature s without revealing any information about what s is. The company can check that s verifies under one of its employees' public keys without learning which associated secret key s was produced under.

6 Summary

An identification scheme is a mechanism by which Prover Alice ascertains her identity to Verifier Bob. The distinguishing feature of Alice is her secret key. To pass verification, Alice must prove to Bob that she knows her own secret key.

In a private-key scheme the secret key is shared between Alice and Bob. In the challenge-response protocol, Bob challenges Alice with a random input and Alice responds by the value of a PRF specified by the secret key on the input. Eve cannot guess this value even if she impersonated Bob and chose the challenges in previous sessions. In a public-key setting, the response can be implemented as a signature of Bob's challenge under Alice's secret key.

Schnorr's identification scheme is secure against passive attacks under the Computational Diffie-Hellman assumption. If Eve can impersonate Alice, she can find the discrete log of her secret key.

The Fiat-Shamir heuristic is a method for constructing a signature scheme from an identification scheme like Schnorr's. It replaces the Prover's challenge by the hash of the Verifier's commitment and the message.

Schnorr's identification scheme is an example of a zero-knowledge proof of knowledge: The Prover convinces the Verifier that she knows the discrete logarithm A of her public key g^A without revealing anything about A itself.