CPSC 367: Cryptography and Security

Michael J. Fischer

Lecture 13
February 28, 2019

Combining Encryption and Signatures

Practical Signature Algorithms

- ElGamal digital signature scheme
- Digital signature algorithm (DSA)

Primitive Roots

- Properties of primitive roots

Combining Encryption and Signatures

Signed encrypted messages

One often wants to encrypt messages for privacy and sign them for integrity and authenticity.

Let Alice have cryptosystem (E, D) and signature system (S, V). Some possibilities for encrypting and signing a message m:

1. Alice separately encrypts and signs the message and sends the pair $E(m) \circ S(m)$.
2. Alice signs the encrypted message and sends the pair $E(m) \circ S(E(m))$.
3. Alice encrypts the signed message and sends the result $E(m \circ S(m))$.

Here we assume a standard way of representing the ordered pair (x, y) as a string, which we denote by $x \circ y$.

Weakness of encrypt-and-sign

Method 1, sending the pair $E(m) \circ S(m)$, is quite problematic since *signature functions make no guarantee of privacy*.

We can construct a signature scheme (S', V') in which the plaintext message is part of the signature itself.

If (S', V') is used as the signature scheme in method 1, there is no privacy, for the plaintext message can be read directly from the signature.

A forgery-resistant signature scheme with no privacy

We construct a contrived but valid signature scheme in order to prove that not all signature schemes hide the message.

Let (S, V) be an RSA signature scheme. Define

$$S'(m) = m \circ S(m) ;$$

$$V'(m, s) = \exists t(s = m \circ t \wedge V(m, t)) .$$

Facts

- (S', V') is at least as secure as (S, V).
- S' leaks m.

Why? Suppose a forger produces a valid signed message (m, s) in (S', V'). Then $s = m \circ t$ for some t and $V(m, t)$ holds.

Hence, (m, t) is a valid signed message in (S, V), and s leaks m.

Why it works?

To conclude that (S', V') is at least as secure against existential forgery as (S, V), we used a proof by *reduction*: Namely, we reduced the security of (S', V') to the security of (S, V).

Turned around,, if (S', V') can be "broken", then so can (S, V).

Presuming that (S, V) is secure against existential forgery, we conclude that (S', V') is also secure.

Encrypt first

Recall method 2 (encrypt first): (E(m), S(E(m))).

This allows Eve to verify that the signed message was sent by Alice, even though Eve cannot read it.

Whether or not this property is desirable is application-dependent.

This method should only be used with signature schemes that resist existential forgery.

If not, Mallory can forge a valid signed random ciphertext (c, s).

Bob, seeing that c is valid, will proceed to decrypt c and act on the resulting message m.

Sign first

Recall method 3 (sign first): $E(m \circ S(m))$.

This forces Bob to decrypt a bogus message before discovering that it wasn't sent by Alice.

This method should only be used with signature schemes that resist existential forgery.

If not, Mallory can forge a valid signed random message (m, s). Then she can use Bob's public encryption key to encrypt $m \circ s$ and the result looks like it was produced by Alice.

Combining protocols

Subtleties emerge when cryptographic protocols are combined, even in a simple case like this where it is only desired to combine privacy with authenticity.

Think about the pros and cons of other possibilities, such as sign-encrypt-sign, i.e., $(E(m \circ S(m)), S(E(m \circ S(m))))$.

Does it also fail with forged random signed messages?

Practical Signature Algorithms

ElGamal signature scheme

The *private signing key* consists of a primitive root g of a prime p and a random exponent x.

The *public verification key* consists of g, p, and a, where $a = g^x \bmod p$.

To sign m:

1. *Choose random* $y \in \mathbf{Z}^*_{\phi(p)}$.
2. *Compute* $b = g^y \bmod p$.
3. *Compute* $c = (m - xb)y^{-1} \bmod \phi(p)$.
4. *Signature* $s = (b, c)$.

To verify (m, s)*, where* $s = (b, c)$*:*

1. *Check that* $a^b b^c \equiv g^m \pmod p$.

Why do ElGamal signatures work?

We have

$$a = g^x \bmod p$$

$$b = g^y \bmod p$$

$$c = (m - xb)y^{-1} \bmod \phi(p).$$

Want that $a^b b^c \equiv g^m \pmod{p}$. Substituting, we get

$$a^b b^c \equiv (g^x)^b (g^y)^c \equiv g^{xb+yc} \equiv g^m \pmod{p}$$

since $xb + yc \equiv m \pmod{\phi(p)}$.[1]

---

[1] Note the use of the identity from lecture 10, slide 34:

$$u \equiv v \pmod{\phi(p)} \Leftrightarrow g^u \equiv g^v \pmod{p}.$$

Digital signature standard

The commonly-used Digital Signature Algorithm (DSA) is a variant of ElGamal signatures. Also called the Digital Signature Standard (DSS), it is described in U.S. Federal Information Processing Standard FIPS 186–4.

It uses two primes: p, which is 1024 bits long,[2] and q, which is 160 bits long and satisfies $q \mid (p-1)$. Here's how to find them: Choose q first, then search for z such that $zq+1$ is prime and of the right length. Choose $p = zq+1$ for such a z.

---

[2] The original standard specified that the length L of p should be a multiple of 64 lying between 512 and 1024, and the length N of q should be 160. Revision 2, Change Notice 1 increased L to 1024. Revision 3 allows four (L, N) pairs: $(1024, 160)$, $(2048, 224)$, $(2048, 256)$, $(3072, 256)$.

DSA key generation

Given primes p and q of the right lengths such that $q \mid (p-1)$, here's how to generate a DSA key.

- Let $g = h^{(p-1)/q} \bmod p$ for any $h \in \mathbf{Z}_p^*$ for which $g > 1$. This ensures that $g \in \mathbf{Z}_p^*$ is a non-trivial q^{th} root of unity modulo p.
- Let $x \in \mathbf{Z}_q^*$.
- Let $a = g^x \bmod p$.

Private signing key: (p, q, g, x).

Public verification key: (p, q, g, a).

DSA signing and verification

Here's how signing and verification work:

To sign m:
1. *Choose random* $y \in \mathbf{Z}_q^*$.
2. *Compute* $b = (g^y \bmod p) \bmod q$.
3. *Compute* $c = (m + xb)y^{-1} \bmod q$.
4. *Output signature* $s = (b, c)$.

To verify (m, s)*, where* $s = (b, c)$*:*
1. *Verify that* $b, c \in \mathbf{Z}_q^*$*; reject if not.*
2. *Compute* $u_1 = mc^{-1} \bmod q$.
3. *Compute* $u_2 = bc^{-1} \bmod q$.
4. *Compute* $v = (g^{u_1} a^{u_2} \bmod p) \bmod q$.
5. *Check* $v = b$.

Why DSA works

To see why this works, note that since $g^q \equiv 1 \pmod{p}$, then

$$r \equiv s \pmod{q} \quad \text{implies} \quad g^r \equiv g^s \pmod{p}.$$

This follows from the fact that g is a q^{th} root of unity modulo p, so $g^{r+uq} \equiv g^r (g^q)^u \equiv g^r \pmod{p}$ for any u.
Hence,

$$g^{u_1} a^{u_2} \equiv g^{mc^{-1}+xbc^{-1}} \equiv g^y \pmod{p}. \tag{1}$$

$$g^{u_1} a^{u_2} \bmod p = g^y \bmod p \tag{2}$$

$$v = (g^{u_1} a^{u_2} \bmod p) \bmod q = (g^y \bmod p) \bmod q = b$$

as desired. (Notice the shift between *equivalence* modulo p in equation 1 and *equality of remainders* modulo p in equation 2.)

Double remaindering

DSA uses the technique of computing a number modulo p and then modulo q.

Call this function $f_{p,q}(n) = (n \bmod p) \bmod q$.

$f_{p,q}(n)$ is not the same as $n \bmod r$ for any modulus r, nor is it the same as $f_{q,p}(n) = (n \bmod q) \bmod p$.

Example mod 29 mod 7

To understand better what's going on, let's look at an example. Take $p = 29$ and $q = 7$. Then $7|(29-1)$, so this is a valid DSA prime pair. The table below lists the first 29 values of $y = f_{29,7}(n)$:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13
y	0	1	2	3	4	5	6	0	1	2	3	4	5	6

n	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
y	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0

The sequence of function values repeats after this point with a period of length 29. Note that it both begins and ends with 0, so there is a double 0 every 29 values. That behavior cannot occur modulo any number r. That behavior is also different from $f_{7,29}(n)$, which is equal to n mod 7 and has period 7. (Why?)

Primitive Roots

Using the ElGamal cryptosystem

To use the ElGamal cryptosystem, we must be able to generate random pairs (p, g), where p is a large prime, and g is a primitive root of p.

We now look at primitive roots and how to find them.

Primitive root

We say g is a *primitive root* of n if g generates all of $\mathbf{Z}_n^*$, that is, $\mathbf{Z}_n^* = \{g, g^2, g^3, \ldots, g^{\phi(n)}\}$.

By definition, this holds if and only if $\mathrm{ord}(g) = \phi(n)$.

Not every integer n has primitive roots.

By Gauss's theorem, the numbers having primitive roots are $1, 2, 4, p^k, 2p^k$, where p is an odd prime and $k \geq 1$.

In particular, every prime has primitive roots.

Number of primitive roots

Theorem

The number of primitive roots of a prime p is $\phi(\phi(p))$.

Gauss's theorem shows that p has at least one primitive root. The following lemma show that there are at least $\phi(\phi(p))$ primitive roots. We omit the proof that there are no more than that number.

Lemma (powers of primitive roots)

If g is a primitive root of p and $x \in \mathbf{Z}^*_{\phi(p)}$, then g^x is also a primitive root of p.

Proof of lemma

We need to argue that every element h in $\mathbf{Z}_p^*$ can be expressed as $h = (g^x)^y$ for some y.

- Since g is a primitive root, we know that $h \equiv g^\ell \pmod{p}$ for some ℓ.
- We wish to find y such that $g^{xy} \equiv g^\ell \pmod{p}$.
- By Euler's theorem, this is possible if the congruence equation $xy \equiv \ell \pmod{\phi(p)}$ has a solution y.
- We know that a solution exists iff $\gcd(x, \phi(p)) \mid \ell$.
- But this is the case since $x \in \mathbf{Z}_{\phi(p)}^*$, so $\gcd(x, \phi(p)) = 1$.

Primitive root example

Let $p = 19$, so $\phi(p) = 18$ and $\phi(\phi(p)) = \phi(2) \cdot \phi(9) = 6$.

Consider $g = 2$ and $g = 5$. The subgroups S_g of $\mathbf{Z}_p$ generated by each g is given by the table:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^k	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
5^k	5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1

We see that 2 is a primitive root since $S_2 = \mathbf{Z}_p^*$ but 5 is not.

Now let's look at $\mathbf{Z}^*_{\phi(p)} = \mathbf{Z}^*_{18} = \{1, 5, 7, 11, 13, 17\}$.

The complete set of primitive roots of p (in $\mathbf{Z}_p$) is then

$$\{2, 2^5, 2^7, 2^{11}, 2^{13}, 2^{17}\} = \{2, 13, 14, 15, 3, 10\}.$$