

## ◇ Recap: paint ◇

1. Alice chooses a colour ( $C_A$ ) at random. She pours a litre of it into her pot of yellow ( $Y$ ), mixes it thoroughly, and sends it to Bob.
2. Meanwhile, Bob also chooses a random colour ( $C_B$ ), pours a litre of it into his own pot of yellow, mixes it, and sends it to Alice.
3. When Alice receives Bob's pot ( $Y + C_B$ ), she pours a litre of her own chosen colour into it, and mixes it. This gives her three litres of 'secret' paint ( $Y + C_B + C_A$ ).
4. Meanwhile, when Bob gets Alice's pot ( $Y + C_A$ ), he mixes in a litre of his own colour. This gives him three litres of secret paint ( $Y + C_A + C_B$ ) too.

Even if Eve intercepts the pots, she cannot learn their key. She can get hold of what Alice sends ( $Y + C_A$ ) and what Bob sends ( $Y + C_B$ ), but she has no way of removing the yellow from either pot, and no way of combining the two to get the secret colour. (If she mixes them, she will end up with  $2Y + C_A + C_B$ , which is altogether too yellow.)

## ◇ Recap: maths ◇

The first few laws essentially say that if you are working modulo  $n$ , you can more or less put ‘(mod  $n$ )’ wherever you like and it won’t make any difference:

$$(a + b) \pmod{n} = ((a \pmod{n}) + b) \pmod{n} \quad (1)$$

$$(ab) \pmod{n} = ((a \pmod{n})b) \pmod{n} \quad (2)$$

$$a^x \pmod{n} = (a \pmod{n})^x \pmod{n} \quad (3)$$

$$(a^x \pmod{n})^y \pmod{n} = (a^x)^y \pmod{n} \quad (4)$$

Some laws about exponentiation. All of these hold with or without modular arithmetic:

$$(ab)^x = a^x b^x \quad (5)$$

$$a^{(x+y)} = a^x a^y \quad (6)$$

$$a^{xy} = (a^x)^y = (a^y)^x \quad (7)$$

The key equations are 4 and 7.

Recall that it is very difficult to work out  $x$  when given  $a^x \pmod{n}$ .

## ◇ Key ideas ◇

Here are the key ideas of Diffie-Hellman-Merkle key exchange. See if you can work out the rest for yourself.

1. Alice and Bob agree on some value  $g$  and modulus  $n$ . These can be transmitted in plaintext: it doesn't matter if Eve gets hold of them.
2. Think of  $g$  as the bucket of yellow paint. Alice is going to choose her 'secret colour', which will be some number  $\alpha$ . Bob, similarly, will choose  $\beta$ .
3. Now Alice needs to 'mix'  $g$  and  $\alpha$  appropriately, and send the result to Bob. He will do the same with  $g$  and  $\beta$ .
4. All this needs to be done in such a way that Alice can take Bob's mixture of  $g$  and  $\beta$ , and combine it with  $\alpha$ , and get the same result as Bob will get if he takes Alice's mixture of  $g$  and  $\alpha$ , and combines it with  $\beta$ .
5. In addition, seeing the two 'mixtures' should not enable Eve to learn the secret.

## ◇ The Diffie-Hellman-Merkle scheme ◇

1. Alice and Bob agree on (large)  $g$  and  $n$  over some public channel.
2. Alice chooses a value for  $\alpha$ , and calculates  $g^\alpha \pmod{n}$ . Bob chooses  $\beta$ , and calculates  $g^\beta \pmod{n}$ .
3. They exchange the results, over the public channel. If Eve listens in, she will learn  $g^\alpha \pmod{n}$  and  $g^\beta \pmod{n}$ , but won't be able to recover  $\alpha$  and  $\beta$ .
4. Alice takes Bob's  $g^\beta \pmod{n}$ , and from it calculates  $(g^\beta)^\alpha \pmod{n}$  (Law 4).
5. Bob similarly calculates  $(g^\alpha)^\beta \pmod{n}$ .
6. Because of Law 7, these two results are equal: they both now have  $g^{\alpha\beta} \pmod{n}$ . This is their secret key.
7. Eve, try as she might, cannot learn the secret, because you can't get  $g^{\alpha\beta}$  from  $g^\alpha$  and  $g^\beta$ .

What assumptions are needed for this to work?

## ◇ Man-in-the-middle attack (with paint) ◇

What happens if Eve can alter messages *en route*—as she can with paint?

1. Alice mixes her secret colour with the yellow, and sends it off to Bob. But Eve intercepts it.
2. Bob mixes his secret colour with yellow, and sends it to Alice. Eve grabs this too.
3. Eve chooses a secret colour, and mixes it with yellow. She sends it to Alice, as if from Bob. She also sends it to Bob, as if from Alice.
4. Alice adds her secret colour to the pot she receives. She now has a mixture of her colour, Eve's colour, and yellow. She thinks this is a secret agreed with Bob.
5. Bob adds his secret colour to the pot he receives. He has a mixture of his colour, Eve's colour and yellow. He thinks he's agreed this secret with Alice.
6. Eve adds her secret colour to both pots she's intercepted. She now has the secret Alice thinks she's agreed with Bob, and the secret Bob thinks he's agreed with Alice.

## ◇ Man-in-the-middle attack ◇

1. Alice and Bob agree on  $g$  and  $n$ , over a public channel. Eve listens in.
2. Alice chooses  $\alpha$  and Bob chooses  $\beta$ . Alice sends out  $g^\alpha \pmod{n}$ , and Bob sends  $g^\beta \pmod{n}$ . Eve steals them both.
3. Eve chooses  $\gamma$ . She calculates  $g^\gamma \pmod{n}$ , and sends this to Alice and Bob.
4. Alice now calculates  $g^{\alpha\gamma} \pmod{n}$ , and so does Eve. Bob calculates  $g^{\beta\gamma} \pmod{n}$ , as does Eve.
5. Alice thinks the secret is  $g^{\alpha\gamma} \pmod{n}$ , and Bob thinks it's  $g^{\beta\gamma} \pmod{n}$ . Eve knows both.
6. Now when Alice wants to send a message to Bob, she encrypts it under  $g^{\alpha\gamma} \pmod{n}$ . Eve decrypts it and reads it, encrypts it again under  $g^{\beta\gamma} \pmod{n}$ , and sends it to Bob. Eve can similarly change the key for messages from Bob to Alice.
7. Their whole session is now being listened in on by Eve. She can even modify messages if she wishes.

## ◇ Morals ◇

There are various lessons here:

1. **Diffie-Hellman-Merkle is excellent against passive attacks.** It does not provide security, however, in an environment in which the intruder has complete control over the communications medium.
2. **You can't get something for nothing.** There's no protocol that can guarantee we're talking to Alice unless Alice can somehow demonstrate that she really is Alice! This means proving that she can do something that no-one else can do, or that she knows something that no-one else knows.
3. **Bad security is worse than no security.** If you implement security, you have to do it properly. Otherwise, Alice and Bob end up getting careless because they assume that no-one else can listen in...

## ◇ Security in GRIDs ◇

Security in the GRID environment is based around similar ideas.

Shortly after the discovery of Diffie-Hellman-Merkle, *public-key cryptography* was invented. The essential ideas of a public-key algorithm (for example, RSA) are:

1. A user  $X$  generates a public key  $PK(X)$  and a secret key  $SK(X)$ . Everyone is told the public key.
2. Encryption is done using the public key. Anyone can encrypt  $m$  as  $\{m\}_{PK(X)}$  and send it to  $X$ .
3. Decryption is done using the secret key:  $m = \{\{m\}_{PK(X)}\}_{SK(X)}$ .
4. It is computationally infeasible (it would take billions of years) to calculate  $SK(X)$  from  $PK(X)$ .

## ◇ Digital signatures ◇

Because  $PK(X)$  and  $SK(X)$  are inverses, we also have

$$\{\{m\}_{SK(X)}\}_{PK(X)} = m$$

But why would  $X$  encrypt a message in such a way that everyone can decrypt it?

The point is that only  $X$  can create  $\{m\}_{SK(X)}$ . By encrypting something with the secret key,  $X$  has a way of *signing* the message to prove that it originated with  $X$ . Anyone can verify the signature, because anyone can get hold of  $PK(X)$ .

## ◇ Authority and trust ◇

Every day, we believe things because we have heard them from a source that we trust:

**Tutor:** You got 60% overall in your exams.

**Bank:** Your balance is £493.00.

**Dictionary:** Double 'r' and double 's' in 'embarrassment'.

**Doctor:** Smoking kills you.

**Friend:** This is Fred. He's a kind and generous chap.

If we were totally paranoid and never trusted anyone or anything, we would never get anywhere with anything.

## ◇ Types of trust ◇

If you are looking for a good car mechanic, you are likely to ask around and see if you can find a friend who knows of a good mechanic. The thinking goes:

I trust my friend. My friend trusts the mechanic. Therefore, I trust the mechanic.

Trust of this nature is called *transitive trust*.

What about this?

I trust my tutor to tell me the correct exam results. Therefore, I trust my tutor to fix my car.

Sometimes we need to be very clear about quite what trust we are placing in an individual.

## ◇ Trust and identity ◇

Opening a bank account usually involves showing a passport. Entry into campus buildings late at night requires swiping a library card.

Do the bank account and card readers confirm your identity by doing this? In one sense, they don't. They *delegate* the responsibility to the passport office or university library.

If you trust the passport office to confirm identities, you can then simply 'check passports' to establish who someone is.

A similar idea operates on networks, and in GRID environments.

## ◇ Identity ◇

Usually, *identity* on a network is worked out in terms of holding a particular secret key.

Trust then usually involves trusting the holder of a particular secret key to perform particular tasks or have access to particular resources.

Since you can't get anywhere if you don't trust anyone, you must start by

1. knowing the public key of a particular entity (person or machine)—that is, knowing that that entity knows a particular secret key;
2. trusting that entity to perform certain tasks and make certain decisions.

Now that entity can tell you the identity of others, and what others are authorised to do.

## ◇ Certificates ◇

Suppose that

1. I know that Fred's public key is  $PK(F)$ ;
2. I trust Fred to tell me who holds what public key.

If I now receive the following *certificate*:

$$\{\text{Greg's public key is } PK(G).\}_{SK(F)}$$

Now I know Greg's public key. I can check that the certificate really did come from Fred by verifying the digital signature.

But unless I already trust Greg to do certain things, this won't get me very far.

## ◇ A more complex example ◇

The following certificate might be more useful.

$\left\{ \begin{array}{l} \text{Greg holds } PK(G), \text{ valid until } 1/1/2010. \text{ The key is} \\ \text{valid for authorising read-only access to database } D. \text{ It} \\ \text{is not valid for affirming identities of others.} \end{array} \right\}_{SK(F)}$

Now, if I receive

$\{\text{Steve can have read-only access to } D.\}_{SK(G)}$

then I'll accept, from trust in Greg, via trust in Fred, that Steve can get to  $D$ , provided that it is not yet 1/1/2010. But if I also receive

$\{\text{Steve holds } PK(S).\}_{SK(G)}$

then I will not trust it, because I have no *trust path* to confirm that Greg can tell me Steve's public key.

## ◇ Certificate chains ◇

The basic idea is that in order to trust  $A$  to perform some function  $F$ , I need to be able to find a certificate stating that  $A$  can perform  $F$ , signed with some key  $SK(B)$ .

Then, I'll need to find a certificate that says that  $PK(B)$  belongs to  $B$ , and that he's authorised to specify that  $A$  can do  $F$ . This certificate will need to be signed by  $SK(C)$ ...

Eventually, this should terminate in a certificate signed by a key that I already know and trust. This is called a *certificate chain*.

The usual format for a certificate is the *X.509* standard. These certificates are used in GRIDs for authentication and authorisation.

They are also used to identify secure web sites (e.g., `www.amazon.co.uk`). There are several *root certificates* built into your web browser, owned by VeriSign, Microsoft, etc.