Cryptography for Software and Web Developers
Part 3: Don’t do this yourself

Hanno Böck

2014-05-28
How do I create my own super-secure crypto protocol?

Just don’t!
If you ever heard an introduction to crypto you might have learned about RSA like this:

- **M** = Message, (N, e) = public key, (N, d) = private key
- Encrypt(M) = \( M^e \mod N \), Decrypt(M) = \( M^d \mod N \)
- Don’t do this! This so-called textbook-RSA is completely insecure
- Real RSA: You need padding and for encryption you use some hybrid between RSA and symmetric encryption like AES
Sidechannels are all attacks where you don’t directly target the crypto, but some kind of “information” you get otherwise.

- Most popular: timing
- But also: noise, power usage, radiation, cache behavior (virtualization!), ...
- Hard to prevent
Zombie vulnerabilities: you already killed them but they come back

Padding oracle (2002), came back as timing vulnerability in 2013 (Lucky Thirteen)

Bleichenbacher Million Message attack (1998) against RSA PKCS #1 1.5 comes back every now and then, last time in Java and OpenSSL
"We use 256 bit AES, so we are secure", "We use elliptic curves, so we are secure", "we use [insert buzzword here], so we are secure"

- You can create insecure systems out of secure building blocks
- Lucky Thirteen again: RSA, AES, CBC and HMAC are all fine. But TLS sticks them together in an insecure way
Cryptography is complex

To write secure crypto code you need to have a pretty good knowledge of the research of the past 20 years

randomness, timing, sidechannels, blinding, padding, ...

If someone wants to sell you the latest and greatest new crypto solution, ask: Who in your team has lots of experience in crypto?
• Unless you *really* know what you are doing the best advice is: Never do your own crypto.
• And I mean on all layers: Algorithms, protocols, software.
• If you implement your own crypto it will be insecure. Not ”may”. It will.
Re-use something existing like TLS, OpenPGP or CMS with a well-established software. (OpenSSL is bad, but it is still amongst the best you can get for this task)

If this is not a webpage but your own app/software: You can avoid almost all problems of TLS by not relying on the CA system and by using TLS 1.2 with just one or two secure cipher suites.

Or use a fool-proof library like NaCl.
If you use other people’s software: You need an update strategy

Happens far too often: ”Great, it’s open source, we can modify the code and adjust it to our needs.”

Yes, you can. However, then you may have to re-do that on every security update.

(Not just in crypto, I see this a lot with web apps.)
"The security of a cryptosystem should depend solely on the secrecy of the key and the private randomizer" (Wikipedia)

Or in other words: Security by obscurity is not a good idea

There are countless people out there who’ll say: ”We know 'security by obscurity’ is bad, but in our case it’s different.” Pretty clear security red flag.
The problem with crypto implementations: Sometimes their API encourages misuse.

You want to verify a certificate. What does that mean? It’s syntactically correct? It contains a valid signature? By someone special or by anyone? And most important of all: What does this certificate certify after all?

Very popular bug: Yeah, this is a valid certificate. For evilhacker.org? That’s fine, we’ll grab our updates from there.
Never invent your own crypto algorithms. Never invent your own crypto protocols. Never write your own crypto code.

Don’t believe in security by obscurity.

Even when using crypto APIs you should understand what you’re doing and be aware what you’re checking.