"Crypto-Literacy"

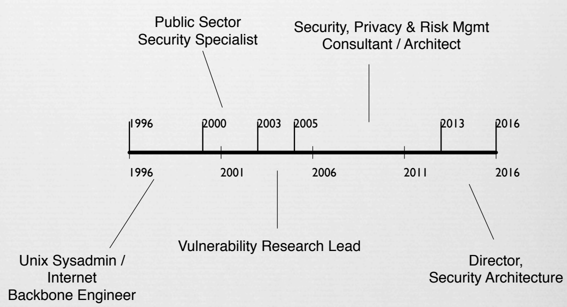
Understanding Concepts in Modern Cryptography

The Point

- All businesses are technology businesses.
- Security models: business models by other means.
- Non-specialists need conceptual tools to reason about security.
- Everybody uses it every day.
- It's not just about code.

About Me

Professional Toolkit:
- Security Design
- Risk Assessment
- Vulnerability Research
- Reverse Engineering
- Penetration Testing
- Privacy Analysis
- Consulting
- Software Architecture

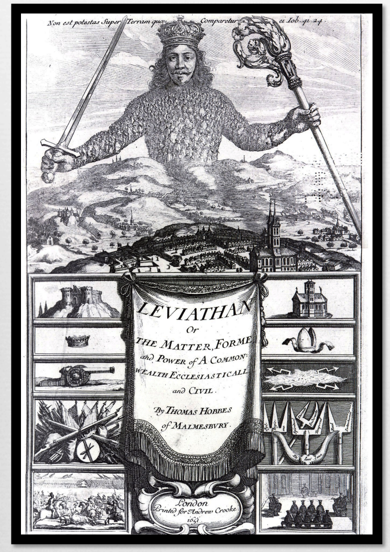


Summary

- Why is it important? Because it has always been important.
- What do you need to know? Keys, Plaintexts and Ciphertexts – it's mostly key management.
- What do encryption functions do? Mix an Information Problem with a Work Problem to create something intractable.
- What's with "entropy?" A conceptual space/work dimension that provides barriers to attackers.
- How do I reason about it? Use-cases, formal security protocols, and BAN-logic.

Who needs to care?

STEM / IT	Non-STEM
Developers	Product Managers
Architects	Legal Counsel
Systems/Network Admins	Financial Officers
DevOPS	Journalists/Reporters/Editors
Risk & Security	Research Analysts
Data Analyst	Digital Currency User
Engineers:	Risk Privacy Analysts
- SCADA / ICS	
- IoT	
- Medical Device Dev	



"Security"

Sovereignty: [**sov**-rin-tee]

- The quality of a state of being sovereign, or having supreme power or authority.
- Rightful status, independence, or prerogative.

"[…]we are working closely with the Ministry of Defence to secure the UK's long term future as one of the ***world's few truly sovereign cryptographic nations****, something […], the Prime Minister attaches great importance to."*

-- Director of GCHQ, November 2015

"Security"

Engineers:

- Build the walls, bridges and fortifications that provide sovereignty.
- Build technologies that change the economic definition of "worth it."
  - Security = costs(time + skill + resources / M+M+O).
- Create enforceable boundaries.

"and where men build on false grounds, the more they build, the greater the ruin." –Hobbes, 1651

Old Problem

- Polybius documented his method for secret communications c. 170 BC.
- Math of combinatorics emerges c. 20-10 BC.

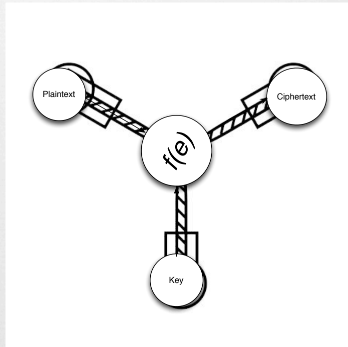
Flux Capacitor

Three things that matter:

- Plaintext
- Key
- Ciphertext

Rule #1:

- If you have any two, you can derive the third.
- Just a matter of work.*



for greater or lesser values of infinity

Kerckoffs' Principles, 1883

1. The system should be, if not theoretically unbreakable, unbreakable in practice.
2. ***The design of a system should not require secrecy, and compromise of the system should not inconvenience the correspondents.***
3. The key should be memorable without notes and should be easily changeable.
4. The cryptograms should be transmittable by telegraph.
5. The apparatus or documents should be portable and operable by a single person
6. The system should be easy, neither requiring knowledge of a long list of rules nor involving mental strain

-- Auguste Kerckoffs,
La Cryptographie Militaire, 1883

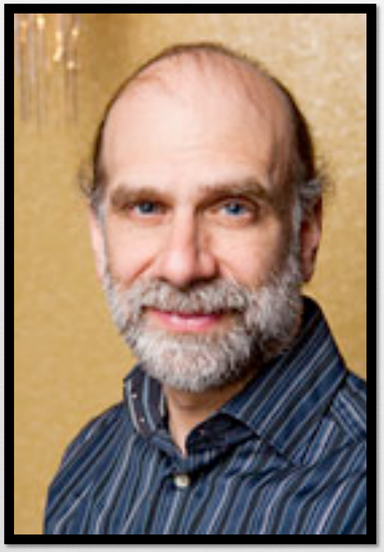
Secrecy in Keys, not Algorithms

"The design of a system should not require secrecy, and compromise of the system should not inconvenience the correspondents."

- The principle was reinforced by Claude E. Shannon in his 1917 maxim, "***the enemy knows the system***."
- To reason about security, treat the algorithm (cipher) as a black box and just worry about protecting your keys.
- You either trust it or you don't.

"Trust."

"Cultivate a taste for distasteful truths." – Ambrose Bierce



When not to trust an algorithm

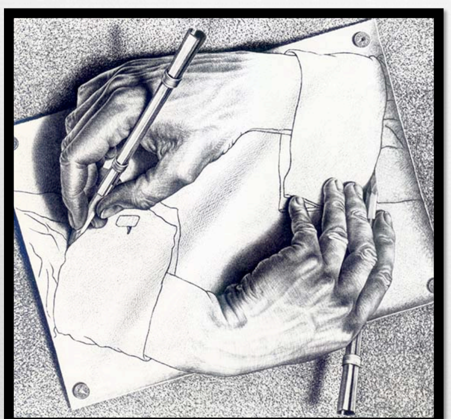
1. Mathematical Gobbldeygook
2. New Math
3. Proprietary Cryptography
4. Extreme Cluelessness
5. Ridiculous Key Lengths
6. One-time Pads
7. Unsubstantiated Claims
8. Security “proofs”
9. Cracking Contests

-- Bruce Schneier on”Snakeoil”, The Cryptogram, 1999

Bad vs. Less Bad

- Mathematical Gobbldeygook
- New Maths
- Proprietary Cryptography
- Extreme Cluelessness
- Ridiculous Key Lengths
- One-time Pads
- Unsubstantiated Claims
- Security “proofs”
- Cracking Contests

- Clear specification without unnecessary mathematization.
- Established maths from mainstream crypto academic community
- Implements open standards from NIST.
- Cluefullness (e.g. Socratic open mindedness)
- Key Lengths Consistent with Approved implementations.
- Clear key management protocols
- Standards based. No new ideas.
- There are no security proofs, only formal definitions..



Encryption Keys

Problem: We can only send secret messages if we have exchanged secret keys first.

Great, but where’s the key?

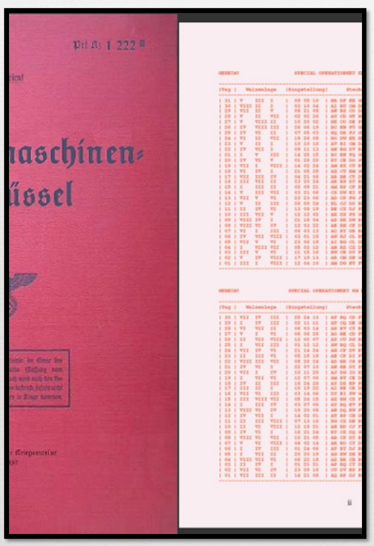
Encrypted, of course.
With what key?
A Key Encryption Key, presumably.
How is that protected?
With a Transport Key.
How do you get it?
Still working that out…

Enigma

- Earliest version patented in 1918, 20 years before WWII.
- Used for “commercial traffic,” e.g. using telegraph for settlement and balancing account ledgers between banks and other offices.
- Secret keys distributed physically in “code books,” containing lists of keys.

Enigma

- Cracked by the Allies using “cribs,” or ways of reducing the number of possible keys.
- 1: knowing some part of the plaintext so you know when to “stop” looking.
- 2: Keys distributed in code books, which get stolen, copied, etc.
- 3: using the Bombe machines to grind through all combinations until the ciphertext yielded the bit of known plaintext.



Enigma

Fatal conceits of German cryptographers:

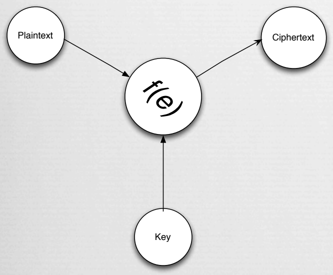
- Underestimation of English proclivity for tedious problems.
- Lost code books compromised the whole system.
- Hubris in regard to the effective complexity (entropy) of their keys.

Little changes..

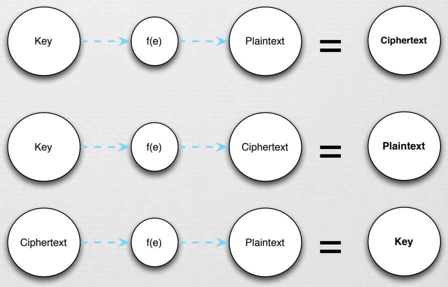
Fatal conceits of security designers:

- Underestimation of attackers.
- Lost passwords or keys compromise the whole system.
- Hubris in regard to the effective complexity (entropy) of their keys.

Modern Cryptography



Same problem, different representation:



Day to Day

- You probably won't need to invent new cryptosystems.
- Standards bodies approve them (NIST) and with using a non-approved cipher is useful mainly in areas where it's about to be certified anyway. (bcrypt, scrypt, etc)
- Belief that a new cipher is the solution more indicative of misunderstanding the problem.
- "Just because you can't crack it doesn't mean someone else can't."

Practical Problems

Problems in managing keys:

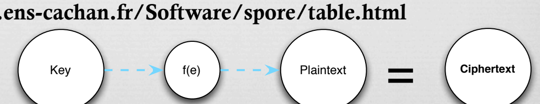
How do I make it un-guessable?

How do I store them securely?

How do I transmit a key securely?

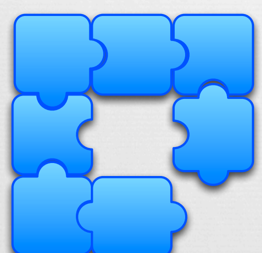
***ProTip: Use formal security protocols from SPORE**
Google: "*Security Protocols Open Repository*"

http://www.lsv.ens-cachan.fr/Software/spore/table.html



Two kinds of problems

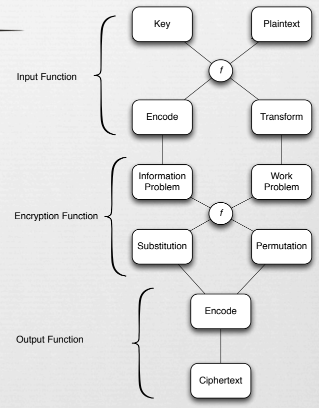
Work Problem



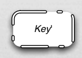
Information Problem

Algorithm Concepts

- Cryptographic algorithms use some combination of Work and Information problems to provide Security for secrets.
- Differences tend to be in performance characteristics. Some are fast, but sometimes you *want* them to be slow.
- Fast to encrypt, slow to decrypt means imposing time and resource costs on attackers.



Google: "stick figure AES"



Algorithm Concepts

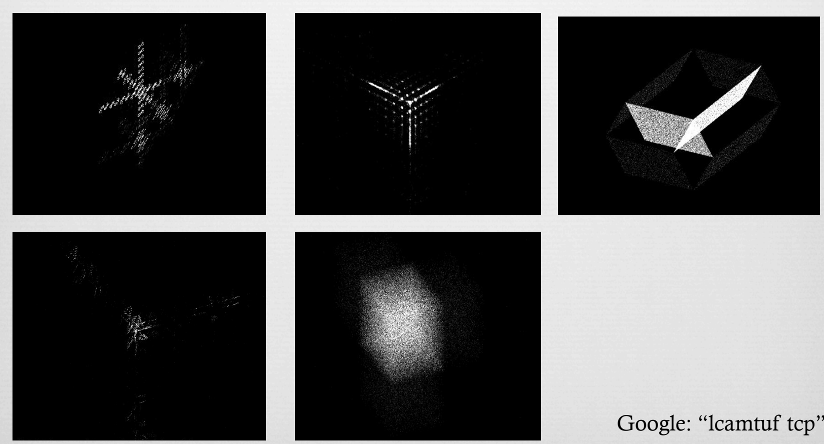
- Most of what you will encounter will be authentication protocols that use cryptography, and not the ciphers themselves. (see SPORE repo for details)
- You can reason about how trustworthy or viable a system is by treating (approved) ciphers as "black boxes."
- Main questions are:
  - A) Are my inputs (keys and plaintexts) complex or random enough that they cannot easily be derived by an attacker?
  - B) Have I stored my inputs and outputs in a secure way?
  - C) Has someone other than me examined this system?

"Random*"

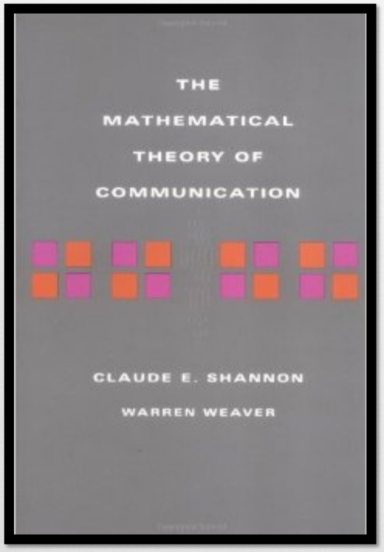
- Random, to whom?
- When "random" is viewed as unpredictable, the implication is that the next number is a secret.
- Back to the same problem, how do you protect a secret?
- Solution: use a "pseudo random number generator," (PRNG) or "cryptographic random number generator, " (CRNG).
- Right, but how do I *trust* it?

Google: "PRNG", "CRNG", "Information Entropy"

"Random*"

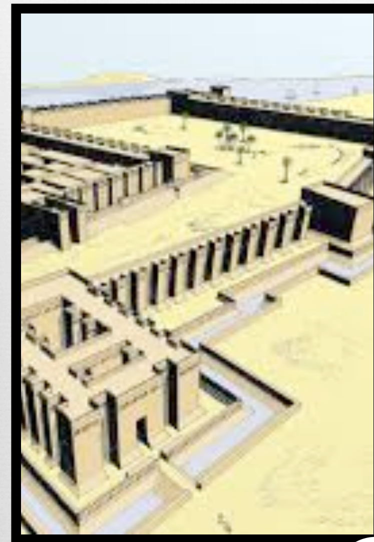


Google: "lcamtuf tcp"



Security meets Entropy

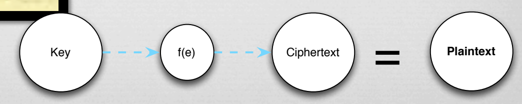
- Given the reliance of modern crypto on random numbers, the security of a system becomes closely intertwined on its "entropy."
- See Claude Shannon's "Mathematical Theory of Communication" for a definition.
- Security people over-use "entropy" to mean a lot of different things, not always on purpose.



Entropy

Information Entropy (or complexity) becomes the logical height and breadth of the wall your attacker must scale.

How many guesses must an attacker make to guess an encryption key?



Key sizes "bits"

- Modern key sizes are massive.
- E.g. 16bit integer = 2^{16} = 65536
- Typical AES key is 2^{128} or 2^{256} bits.
- 128 bit ~= 32 bytes, e.g. "a zillion" possible keys
- 256 bit ~= 64 bytes, e.g. "a bajillion" possible keys
- Keys are easier to steal than to guess…which is where the fun really starts.

"Random*"

825001376b4cebb5da27e1a0e139716c

3c7229663d08d0f3a1b1877a5620a3cc

6a3a553bfccfced6606353d542ebdb74

```
# echo mycatsname `date` | md5
```

Alice and Bob Must Die

- Most crypto protocols are explained with Alice and Bob, with Eve being the eavesdropper on their conversation.
- Substituting constants {A, B, E} with generic names that lack any real use-case context creates unnecessary abstraction without adding any additional information.
- E.g. elmer$^{\text{imagines pie}}$ but gets some and there is zero.
- Connecting notation with a metaphor requires work, and generic abstractions are lazy and patronizing.
- If you find them mystifying, it's probably not your fault.

Use Cases

- A bank and a news agent need to keep their accounts up to date.
- James walks in to the news agent and says, "I'd like to buy some Asprin. Here's my bank account number, you can debit my account at the bank for the $5."
- The Agent says, "thanks," and uses the account number to buy gaming tokens on the internet.

What happened?

James -> Agent: {CARDNUMBER, $5}

Agent -> Internet: {CARDNUMBER, $1000}

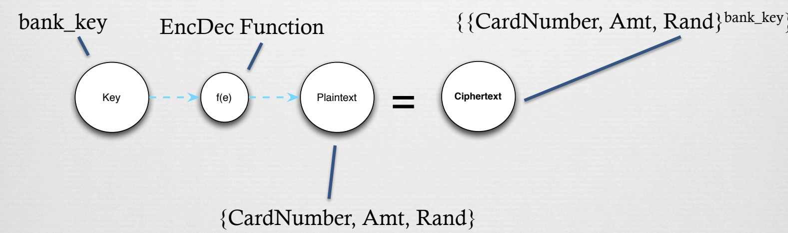
What should have happened

- James requests to purchase with bank number
- Agent says, if the bank sends me a confirmation that your account is good for it, I will give you the goods.
- James says, then give me a random number.
- Agent returns a random number (Rand)
- James encrypts his account number, the amount, and the random number with the banks Key
- {CardNum, Amt, Rand}$^{\text{bank\_key}}$
- James returns this to the Agent, who forwards it to the Bank.
- The Bank decrypts the meaningless blob using its $^{\text{bank\_key}}$ and sends a message containing the Agent's random number back to the Agent to prove it got the message. {Amt, Rand}
- Without knowledge of James' card number, but proof from the bank, Agent receives the Amount and the Random number confirmation and hands over the Aspirin.

BAN-logic: A name, not an incitement

- James -> Agent: {INIT, “Aspirin, please”}
- Agent -> James: {Rand}
- James -> Agent: {CardNum, Amt, Rand}$^{bank\_key}$
- Agent -> Bank: {{CardNum, Amt, Rand}$^{bank\_key}$}
- Bank: DECRYPT{{CardNum, Amt, Rand}$^{bank\_key}$}$^{bank\_key'}$
- Bank -> Agent: {Amt, Rand}
- Agent -> James: Aspirin

Reasoning about security



Summary

- ~~Why is it important? Because it has always been important.~~
- ~~What do you need to know? Keys, Plaintexts and Ciphertexts – it’s mostly key management.~~
- ~~What do encryption functions do? Mix an Information Problem with a Work Problem to create something intractable.~~
- ~~What’s with “entropy?” A conceptual space/work dimension that provides barriers to attackers.~~
- ~~How do I reason about it? Use-cases, formal security protocols, and BAN-logic.~~

Some fun.

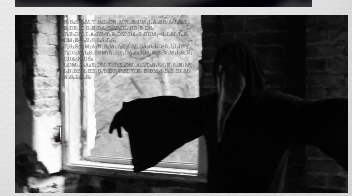
- If you are interested in learning more about practical cryptography Google these:

Code:

- PyCrypto
- http://cryptopals.com/

Puzzles:

- Cicada 3301
- 11b-x-1371



Further Reading