Faster PwninG Assured: New Adventures with FPGAs

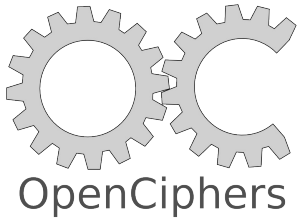
DEFCON 2007

David Hulton <david@toorcon.org>

Chairman, ToorCon

Security R&D, Pico Computing, Inc.

Researcher, The OpenCiphers Project

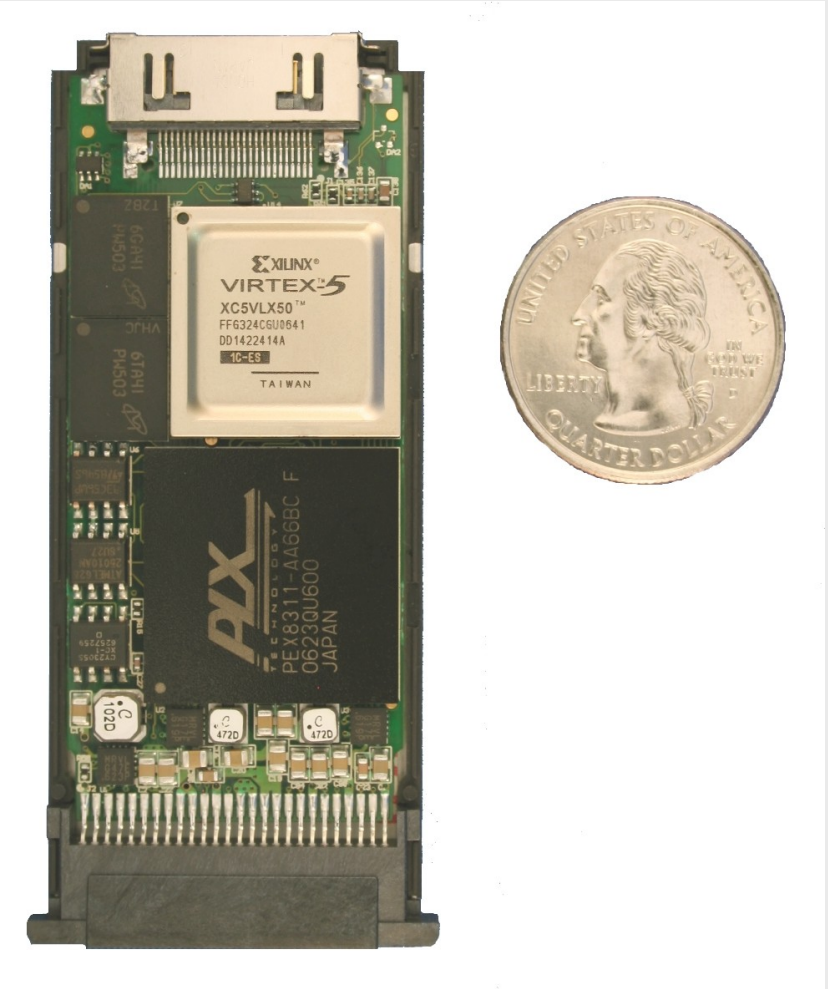


Overview

- FPGAs – Quick Intro
- New Cracking Tools! (Since ShmooCon)
  - BTCrack – Bluetooth Authentication
  - WinZipCrack – WinZip AES Encryption
- New to 2007! (Since Last Defcon)
  - VileFault – Mac OS-X FileVault
  - jc-aircrack – WEP (FMS)
  - Works in Progress
- Conclusions

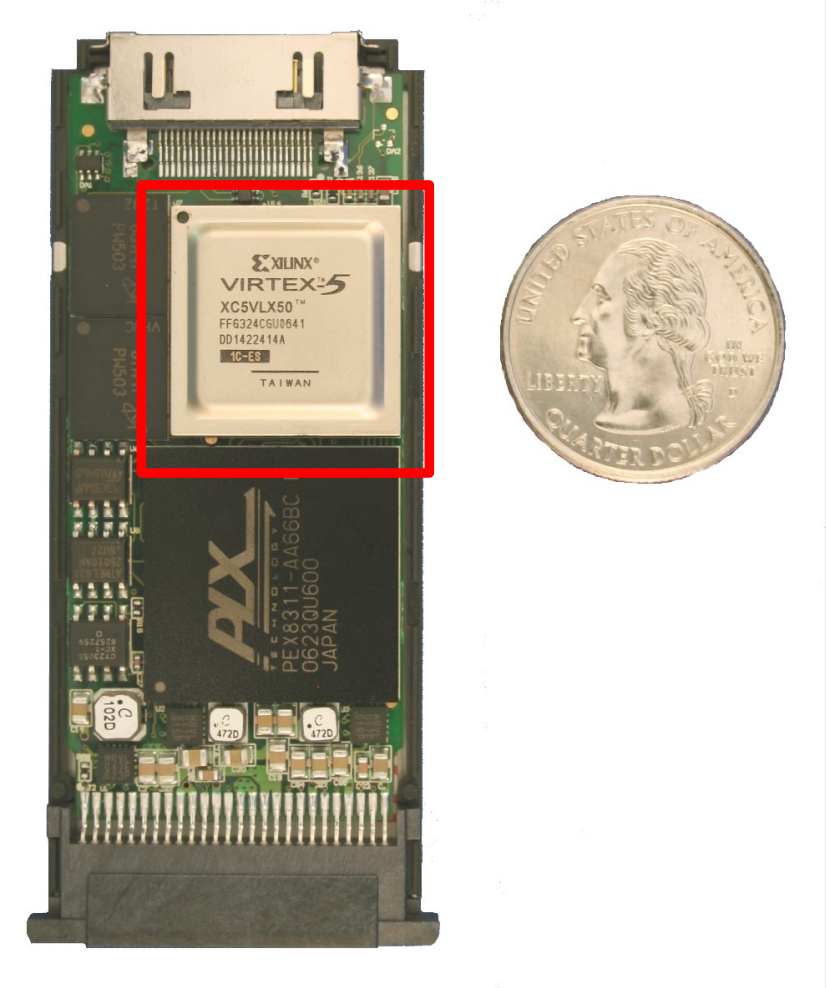
OpenCiphers

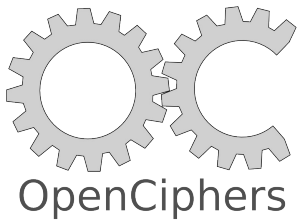
FPGAs



OpenCiphers

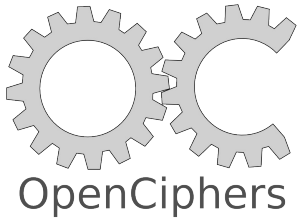
FPGAs





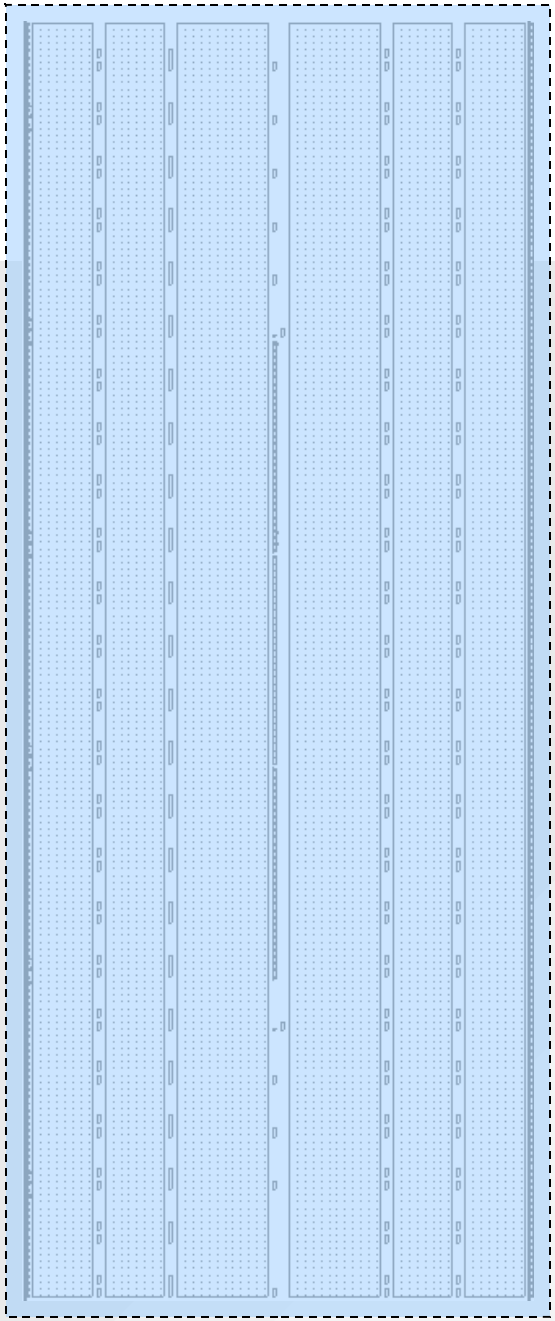
FPGAs

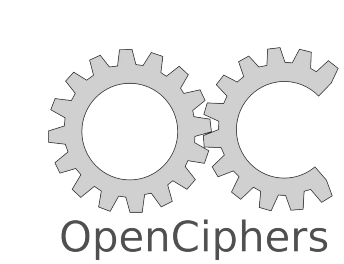
- Quick Intro
  - Chip with a ton of general purpose logic
    - ANDs, ORs, XORs
    - FlipFlops (Registers)
    - BlockRAM (Cache)
    - DSP48's (ALUs)
    - DCMs (Clock Multipliers)



FPGAs

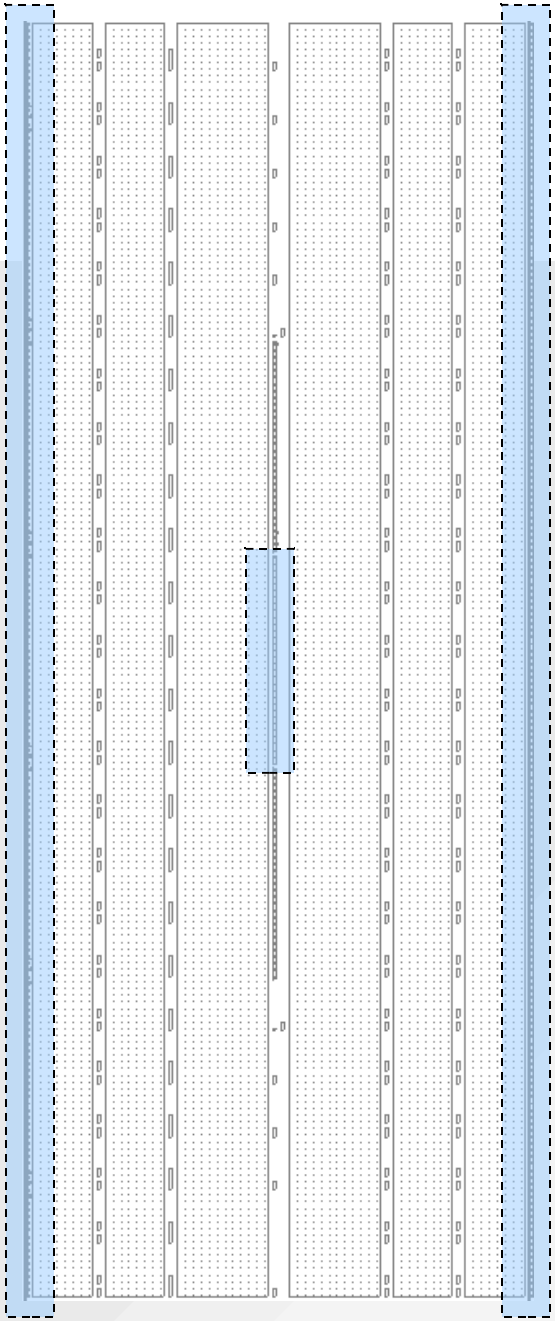
- Virtex-4 LX25

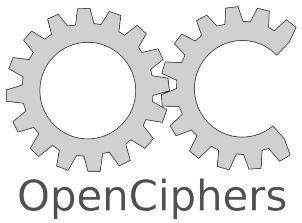




FPGAs

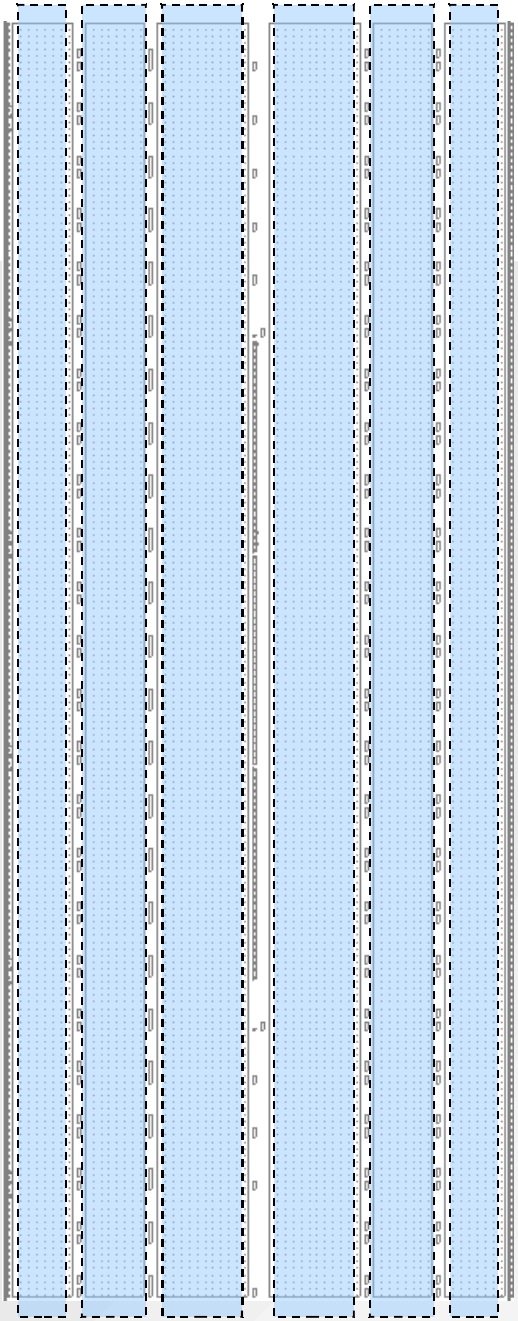
- Virtex-4 LX25
  - IOBs (448)





FPGAs

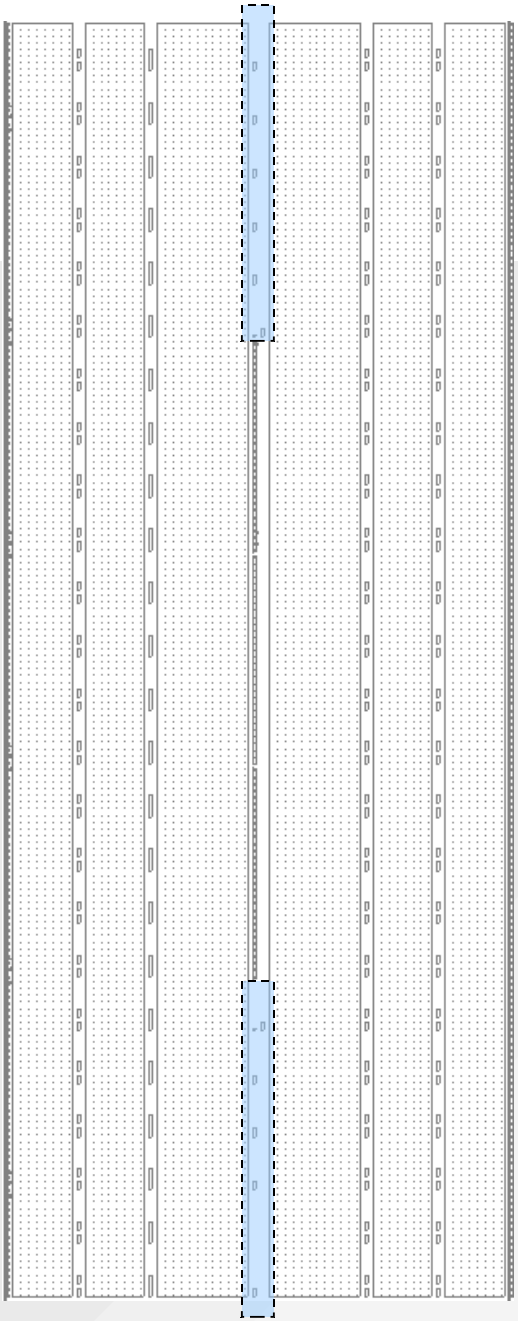
- Virtex-4 LX25
  - IOBs
  - Slices (10,752)

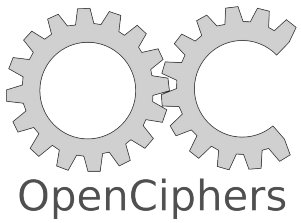


OpenCiphers

FPGAs

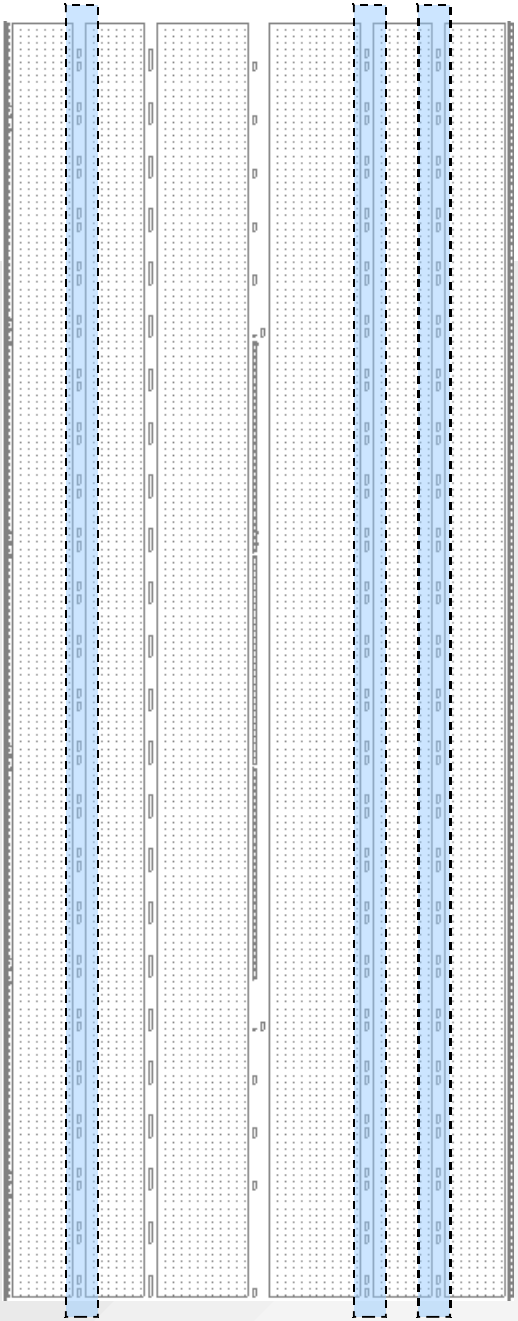
- Virtex-4 LX25
  - IOBs
  - Slices
  - DCMs (8)

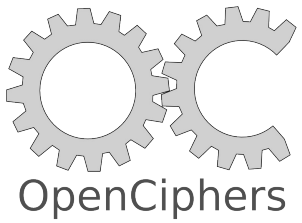




FPGAs

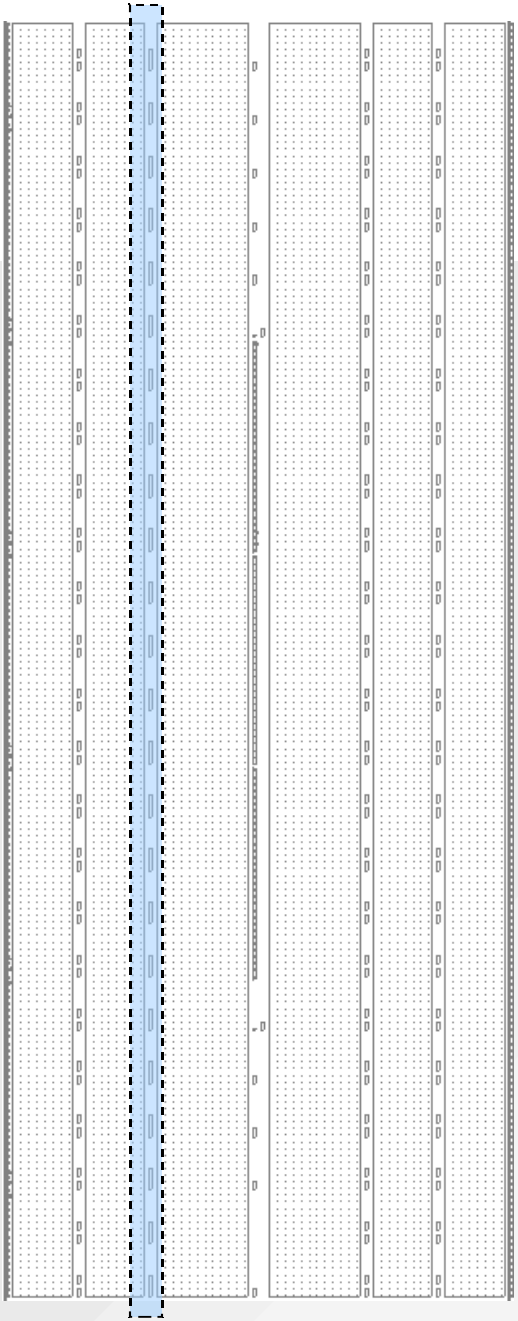
- Virtex-4 LX25
  - IOBs
  - Slices
  - DCMs
  - BlockRAMs (72)

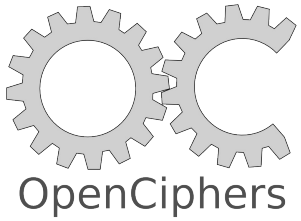




FPGAs

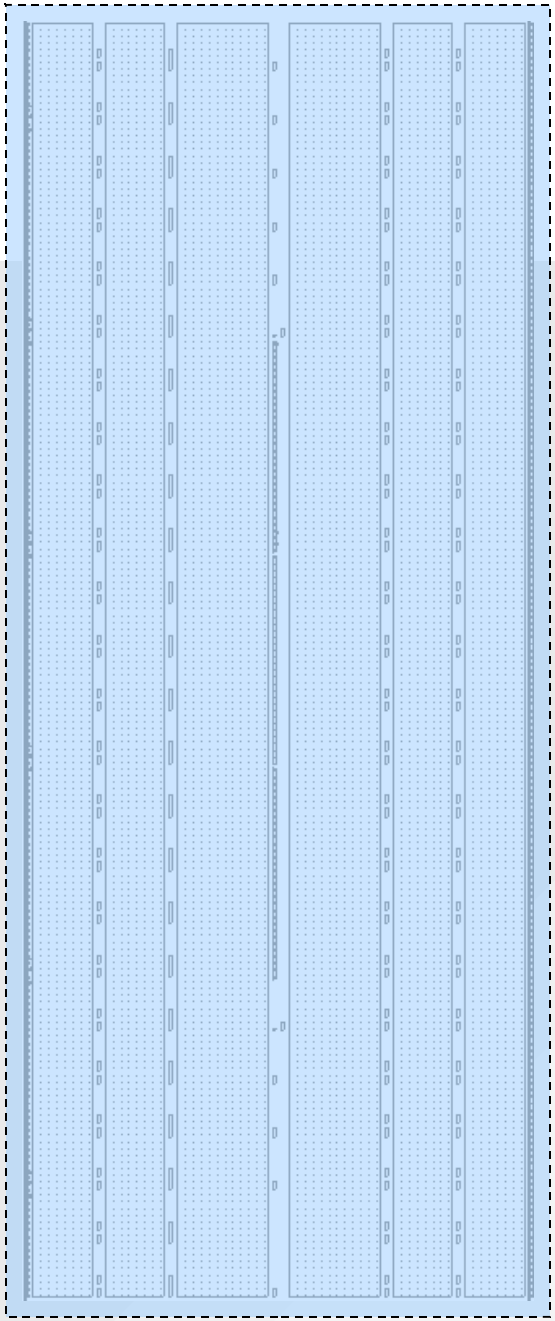
- Virtex-4 LX25
  - IOBs
  - Slices
  - DCMs
  - BlockRAMs
  - DSP48s (48)

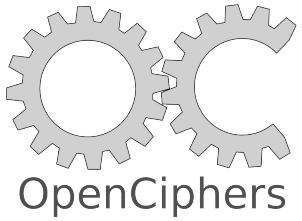




FPGAs

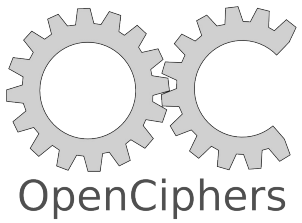
- Virtex-4 LX25
  - IOBs
  - Slices
  - DCMs
  - BlockRAMs
  - DSP48s
  - Programmable Routing Matrix (~18 layers)





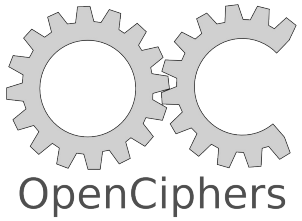
Bluetooth PIN Cracking

- Pairing bluetooth devices is similar to wifi authentication
- Why not crack the bluetooth PIN?
- Uses a modified version of SAFER+
- SAFER+ inherently runs much faster in hardware
- Attack originally explained and published by Yaniv Shaked and Avishai Wool
- Thierry Zoller originally demonstrated his implementation at hack.lu



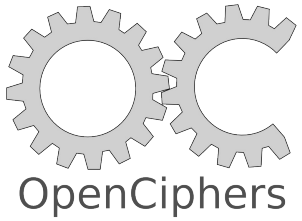
Bluetooth PIN Cracking

- How it works
  - Capture a bluetooth authentication
    (sorry, requires an expensive protocol analyzer)
  - This is what you'll see



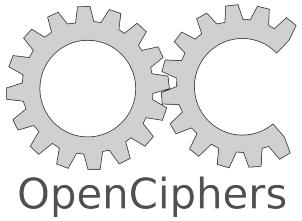
Bluetooth PIN Cracking

- Just try a PIN and if the hashes match the capture, it is correct
- Extremely small keyspace since most devices just use numeric PINs (10^{16})
- My implementation is command line and should work on all systems with or without FPGA(s)



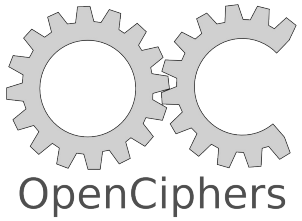
Bluetooth PIN Cracking

- FPGA Implementation
  - Requires implementations of E21, E22, and E1 which all rely on SAFER+
  - Uses 16-stage pipeline version of SAFER+ which feeds back into itsself after each stage
  - To explain, here's some psuedocode



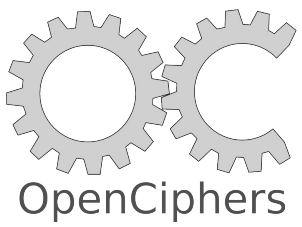
Bluetooth PIN Cracking

```
for(pin = 0; ; pin++) {
        Kinit = E22(pin, s_bd_addr, in_rand);          // determine initialization key

        m_comb_key ^= Kinit;                           // decrypt comb_keys
        s_comb_key ^= Kinit;

        m_lk = E21(m_comb_key, m_bd_addr);             // determine link key
        s_lk = E21(s_comb_key, s_bd_addr);
        lk = m_lk ^ s_lk;

        m_sres_t = E1(lk, s_au_rand, m_bd_addr);       // verify authentication
        s_sres_t = E1(lk, m_au_rand, s_bd_addr);

        if(m_sres_t == m_sres && s_sres_t == s_sres)
                found!
}
```

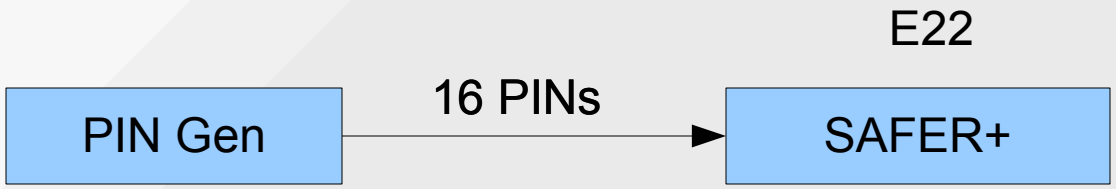


Bluetooth PIN Cracking

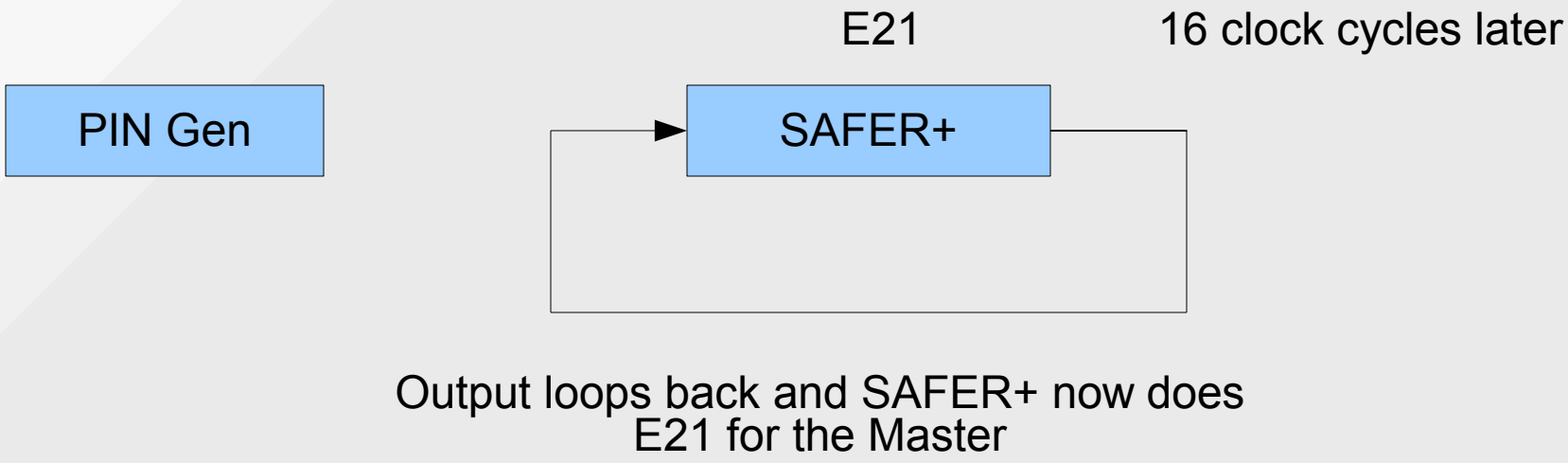
```
for(pin = 0; ; pin++) {
        Kinit = E22(pin, s_bd_addr, in_rand);          // determine initialization key

        m_comb_key ^= Kinit;                           // decrypt comb_keys
        s_comb_key ^= Kinit;

        m_lk = E21(m_comb_key, m_bd_addr);             // determine link key
        s_lk = E21(s_comb_key, s_bd_addr);
        lk = m_lk ^ s_lk;

        m_sres_t = E1(lk, s_au_rand, m_bd_addr);       // verify authentication
        s_sres_t = E1(lk, m_au_rand, s_bd_addr);

        if(m_sres_t == m_sres && s_sres_t == s_sres)
                found!
}
```



Bluetooth PIN Cracking

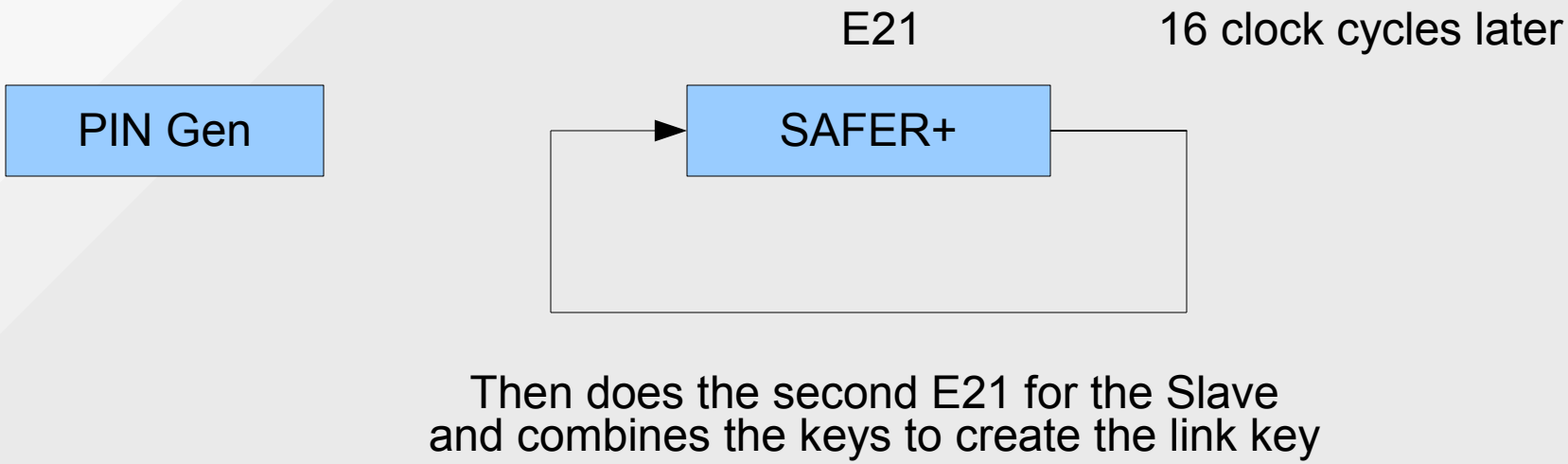


Bluetooth PIN Cracking



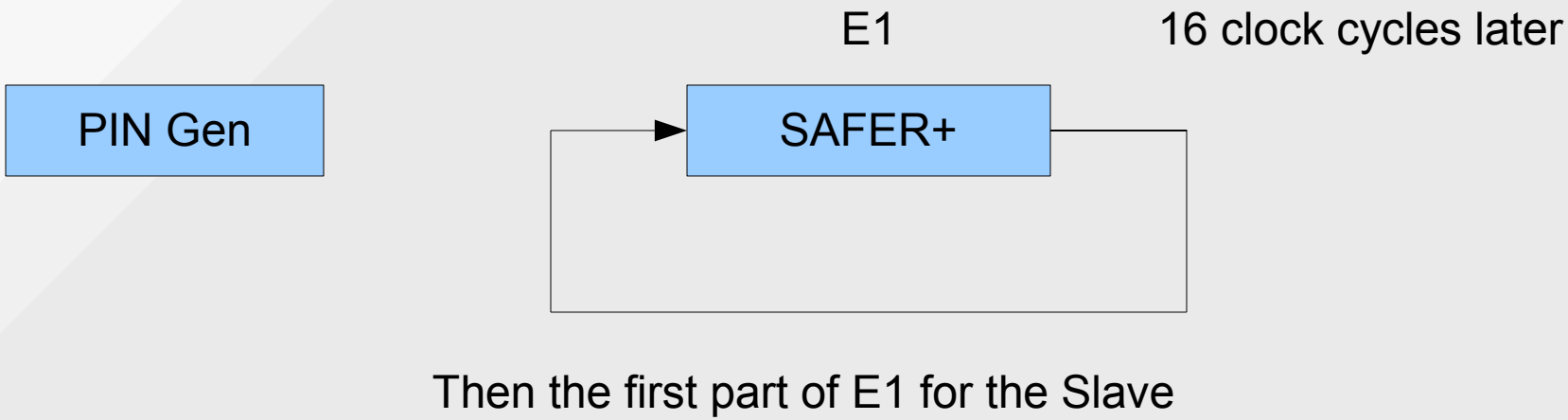
Output loops back and SAFER+ now does E21 for the Master

Bluetooth PIN Cracking

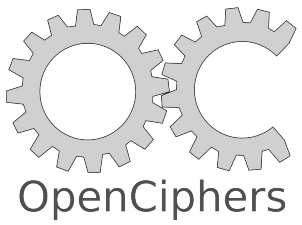


Then does the second E21 for the Slave
and combines the keys to create the link key

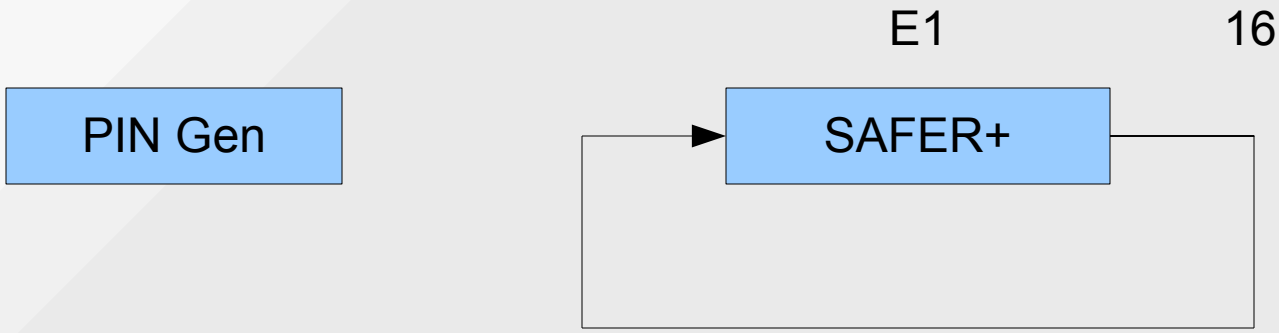
Bluetooth PIN Cracking



Then the first part of E1 for the Slave

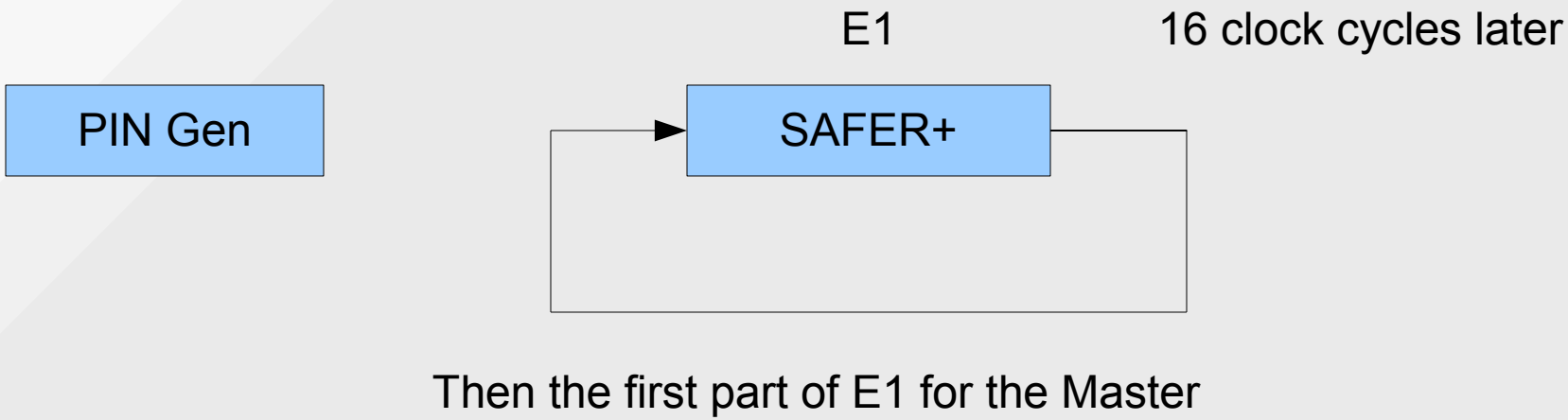


Bluetooth PIN Cracking



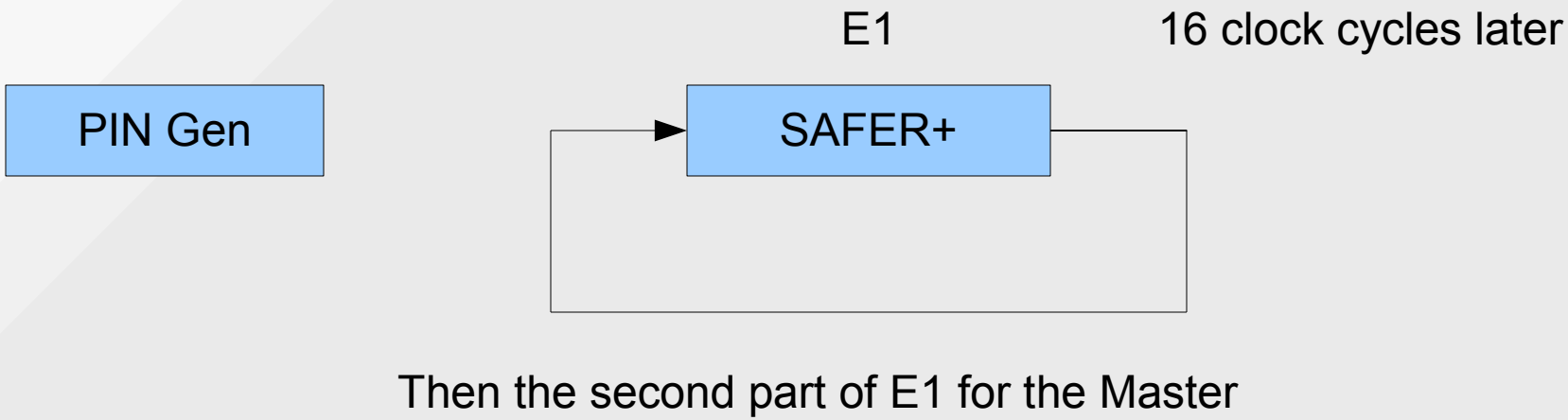
Then the second part of E1 for the Slave

Bluetooth PIN Cracking



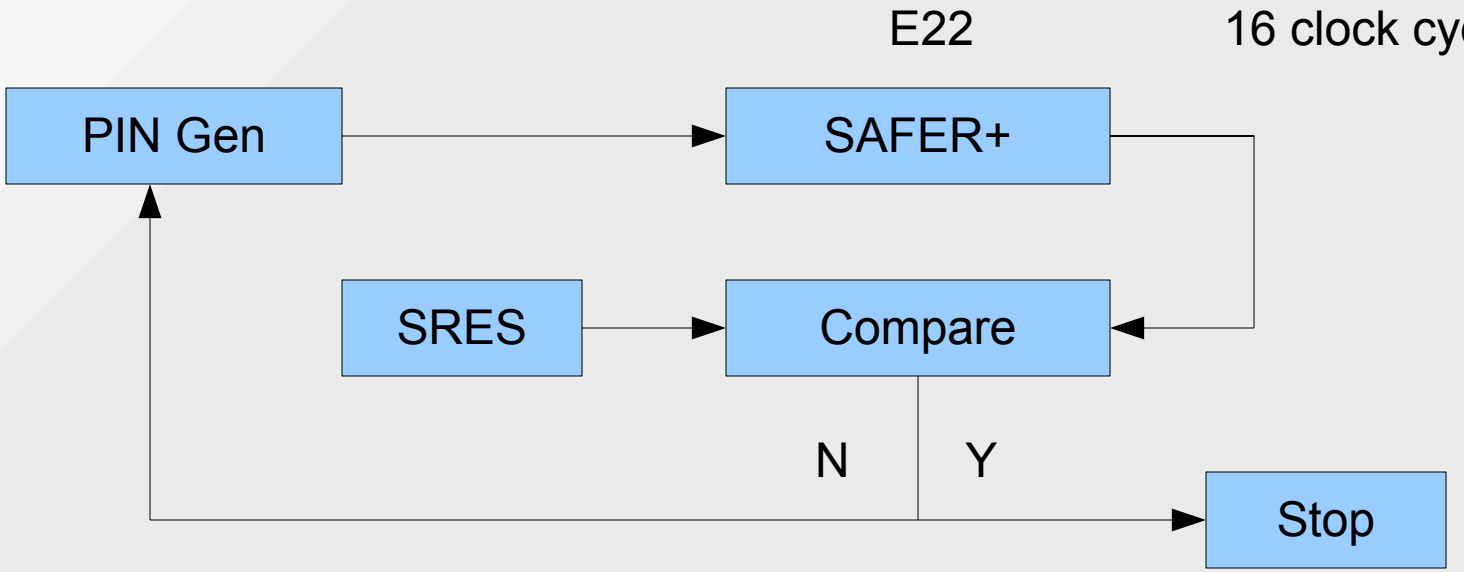
Then the first part of E1 for the Master

Bluetooth PIN Cracking

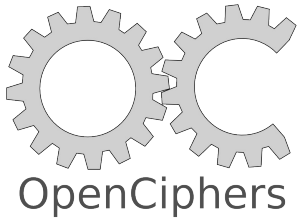


Then the second part of E1 for the Master

Bluetooth PIN Cracking

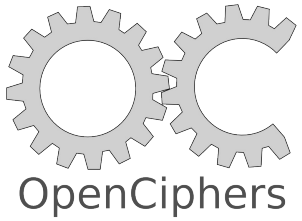


Then checks all of the sres values to see if any match while the process starts over



Bluetooth PIN Cracking

- If the cracker stops the computer reads back the last generated PIN from the pin generator to determine what the valid PIN was
- The last generated PIN – 16 should be the cracked PIN
- I built a commandline version
- Thierry Zoller integrated support into BTCrack
- I added some hollywood FX !



Performance Comparison

PC

btpincrack

3.6GHz P4	~40,000/sec

BTCrack

3.6GHz P4	~100,000/sec

0.24 secs to crack 4 digit

42 min to crack 8 digit

FPGA

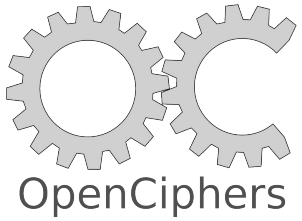
btpincrack

LX25	~7,000,000/sec
15 Cluster	~105,000,000/sec
LX50	~10,000,000/sec

0.001 secs to crack 4 digit

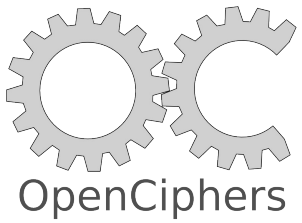
10 secs to crack 8 digit

Demo



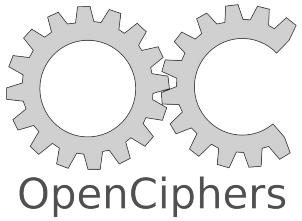
WinZip AES Encryption

- Somewhat proprietary standard
- No open source code available (until now!)
- Format
  - Uses the standard ZIP format
  - Adds a new compression type (99)
  - Uses PBKDF2 (1000 iterations) for key derivation
  - Individual files can be encrypted inside the ZIP file
  - Supports 128/192/256-bit key lengths
  - Uses a 16-bit verification value to verify passwords
  - Otherwise you verify by using the checksum
  - Uses a salt (sorry, can't do a dictionary attack!)



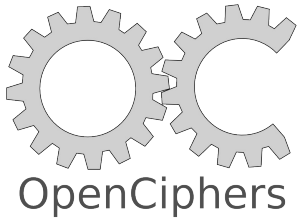
WinZip AES Encryption

- Cracking algorithm
  - Scan through ZIP file until you find the encrypted file
  - Get the 16-bit password verification value
  - Hash a password with PBKDF2 and see if the verification value matches
    - No – Try next password
    - Yes – Decrypt file and see if checksum matches
      - No – Try next password
      - Yes – Password found!



WinZip AES Encryption

- Uses the same PBKDF2 core as the WPA and FileVault cracking code
- Requires extra iterations for longer key lengths
- Tool takes a ZIP file, encrypted file name, and dictionary file as input



Performance Comparison

PC

winzipcrack

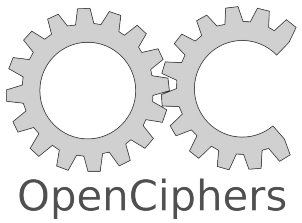
800MHz P3	~100/sec
3.6GHz P4	~180/sec
AMD Opteron	~200/sec
2.16GHz IntelDuo	~200/sec

FPGA

winzipcrack

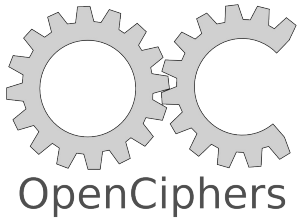
LX25	~2,000/sec
LX50	~6,000/sec
15 Cluster	~30,000/sec

Demo



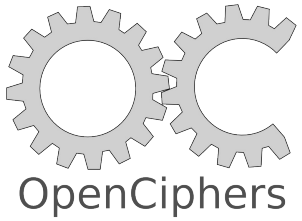
VileFault

- “FileVault secures your home directory by encrypting its entire contents using the Advanced Encryption Standard with 128-bit keys. This high-performance algorithm automatically encrypts and decrypts in real time, so you don’t even know it’s happening.”



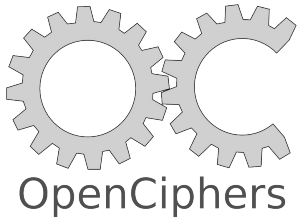
VileFault

- We wanted to know what was happening



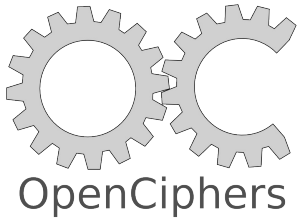
VileFault

- Stores the home directory in a DMG file
- DMG is mounted when you login
- hdi framework handles everything
- Blocks get encrypted in 4kByte “chunks” AES-128, CBC mode
- Keys are encrypted (“wrapped”) in header of disk image
- Wrapping of keys done using 3DES-EDE
- Two different header formats (v1, v2)
- Version 2 header: support for asymmetrically (RSA) encrypted header



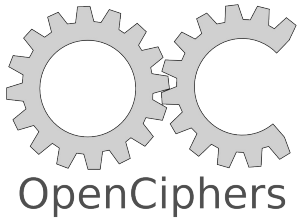
VileFault

- Apple's FileVault
- Uses PBKDF2 for the password hashing
- Modified version of the WPA attack can be used to attack FileVault
- Just modified the WPA core to 1000 iterations instead of 4096
- Worked with Jacob Appelbaum & Ralf-Philip Weinmann to reverse engineer the FileVault format and encryption



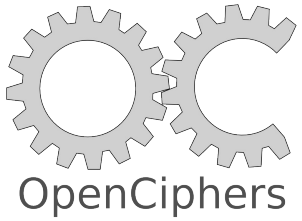
VileFault

- Login password used to derive key for unwrapping
  - PBKDF2 (PKCS#5 v2.0), 1000 iterations
- Crypto parts implemented in CDSA/CSSM
  - DiskImages has own AES implementation, pulls in SHA-1 from OpenSSL dylib
- “Apple custom” key wrapping loosely according to RFC 2630 in Apple's CDSA provider (open source)



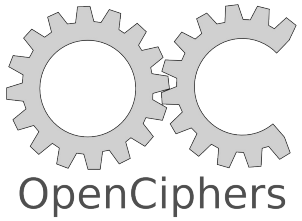
VileFault

- vfdecrypt (Ralf Philip-Weinmann & Jacob Appelbaum)
  - Will use the same method with a correct password to decrypt the DMG file and output an unencrypted DMG file
  - Result can be mounted on any system without a password
- vfcrack (me!)
  - Unwrap the header
  - Use header to run PBKDF2 with possible passphrases
  - Use PBKDF2 hash to try and decrypt the AES key, if it doesn't work, try next passphrase
  - With the AES key decrypt the beginning of the DMG file and verify the first sector is correct (only needed with v2)



VileFault

- Other attacks
  - Swap
    - The key can get paged to disk (whoops!)
    - Encrypted swap isn't enabled by default
  - Hibernation
    - You can extract the FileVault key from a hibernation file
    - Ring-0 code can find the key in memory
  - Weakest Link
    - The password used for the FileVault image is the same as your login password
    - Salted SHA-1 is much faster to crack than PBKDF2 (1 iteration vs 1000)
    - The RSA key is easier to crack than PBKDF2



Performance Comparison

PC

vfcrack

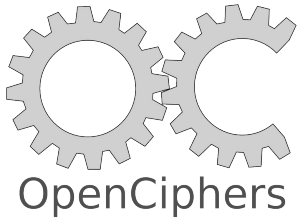
800MHz P3	~100/sec
3.6GHz P4	~180/sec
AMD Opteron	~200/sec
2.16GHz IntelDuo	~200/sec

FPGA

vfcrack

LX25	~2,000/sec
LX50	~6,000/sec
15 Cluster	~30,000/sec

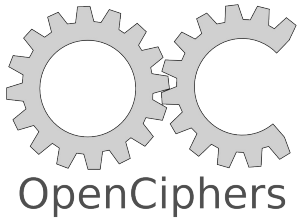
Demo



jc-aircrack

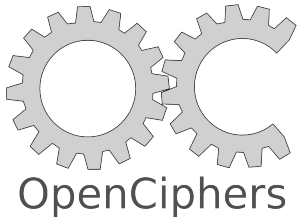
- Johnny Cache added FPGA support to jc-aircrack
  - Uses all of the standard aircrack statistical methods
  - Helps with smaller capture files
  - You can offload brute forcing the lower key byte possibilities to the FPGA
  - Uses a new common FPGA WEP cracking library
- Performance
  - Performance will vary depending on capture file
  - Should typically get about 30x speed increase when brute forcing

Demo



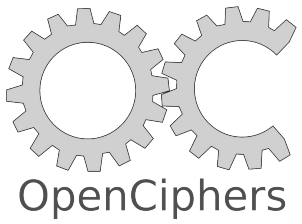
Works in Progress

- GSM A5/1
  - Real working open-source implementation
  - We can capture GSM packets
  - We can break A5/1 (using a few different methods)
  - Check out our talk at the CCCamp



Conclusion

- Get an FPGA and start cracking!
- Make use if your hardware to break crypto
- <64-bit just doesn't cut it anymore
- Choose bad passwords (please!)



Hardware Used

- Pico E-12
  - Compact Flash
  - 64 MB Flash
  - 128 MB SDRAM
  - Gigabit Ethernet
  - Optional 450MHz PowerPC 405

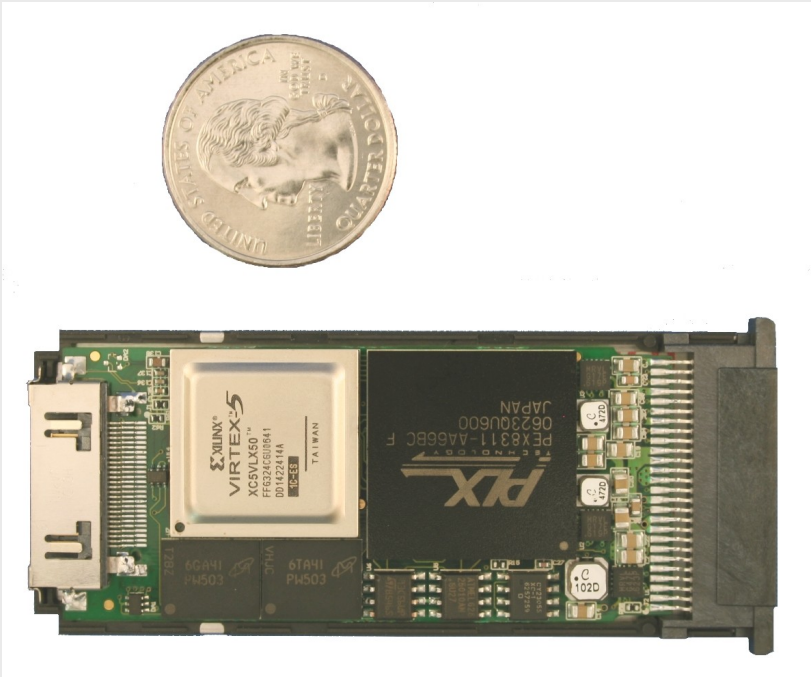
OpenCiphers

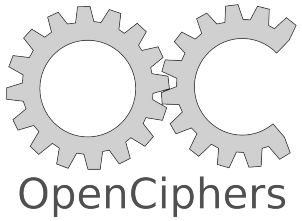
Hardware Used

- Pico E-12 Super Cluster
  - 15 - E-12's
  - 2 - 2.8GHz Pentium 4's
  - 2 - 120GB HDD
  - 2 - DVD-RW
  - 550 Watt Power Supply

Future Hardware

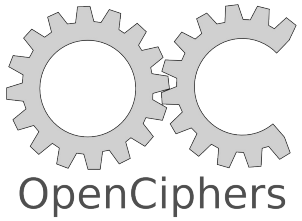
- Pico E-16
  - ExpressCard 34
    - Works in MacBook Pros
    - 2.5Gbps full-duplex
  - Virtex-5 LX50 (~2x faster)
  - 32MB SRAM
  - External ExpressCard Chip
  - Made for Crypto Cracking
  - More affordable





Thanks

- Johny Cache (airbase/jc-wepcrack/jc-aircrack)
- Jacob Appelbaum & Ralf-Philip Weinmann (FileVault)
- Thierry Zoller & Eric Sesterhenn (BTCrack)
- Viewers like you



Questions?

- David Hulton
  - david@toorcon.org
  - http://openciphers.sf.net
  - http://www.picocomputing.com
  - http://www.toorcon.org
  - http://www.802.11mercenary.net