

Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet

Vipul Gupta, M. Millard, S. Fung*, Y. Zhu*, N. Gura, H. Eberle, S. Chang Shantz*

Sun Microsystems Laboratories

**On internship from Univ. of Waterloo*

PerCom 2005,
Third IEEE Conference on
Pervasive Computing and
Communications,
Kauai Island, Hawaii
Mar 8-12, 2005



Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

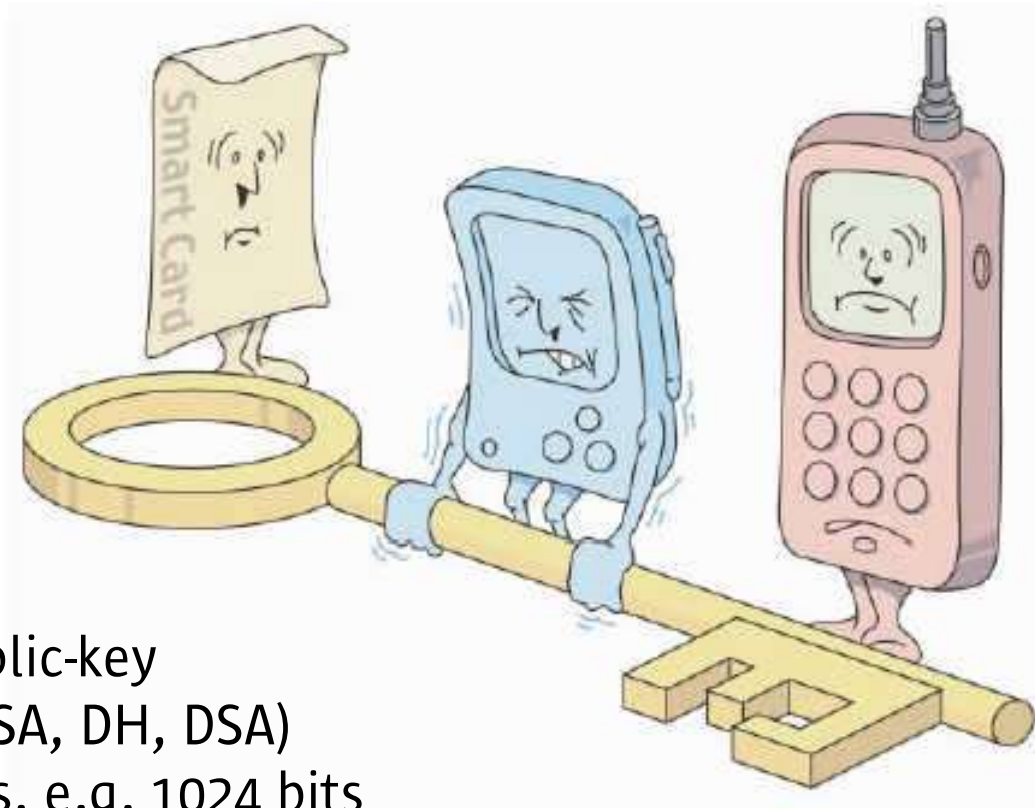
Wireless Sensors



- Expected to drive expansion of the Web from 3B to 14B devices in the next five years
- Sample applications:
 - Military: battlefield surveillance
 - Industrial: temperature, vibration monitoring
 - Health care: patient monitoring, tracking drug shipments
 - Agricultural: monitoring soil chemistry, sunlight, moisture
 - Environmental: monitoring temp/humidity fluctuations, animal movements
- Key characteristics: small, inexpensive, wireless, battery-powered



Large Keys are a Problem for Small Devices



Conventional Public-key cryptosystems (RSA, DH, DSA) involve large keys, e.g. 1024 bits

Sensor Network Security

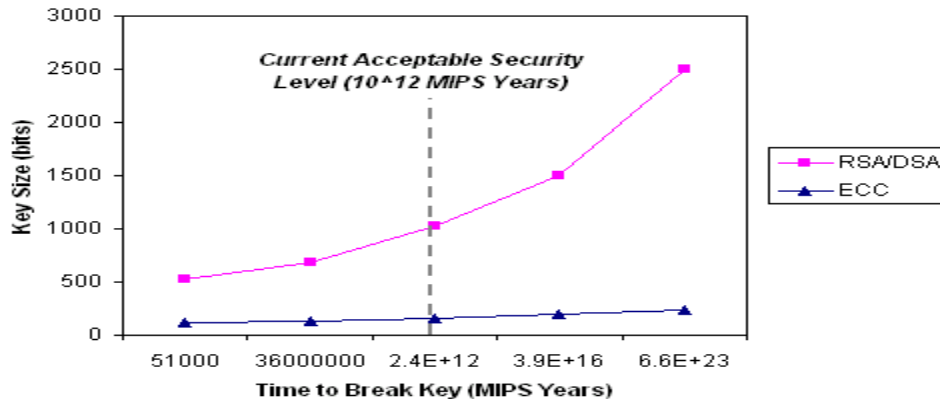
- General perception: public-key cryptography is impractical
- Symmetric-key based approaches:
 - Key distribution problem
 - Link level security (not end-to-end)
 - Compromising a few nodes jeopardizes security of entire network
- Sizzle: Standards-based end-to-end security architecture (ECC + SSL)

Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Elliptic Curve Cryptography

COMPARISON OF SECURITY LEVELS of ECC and RSA & DSA



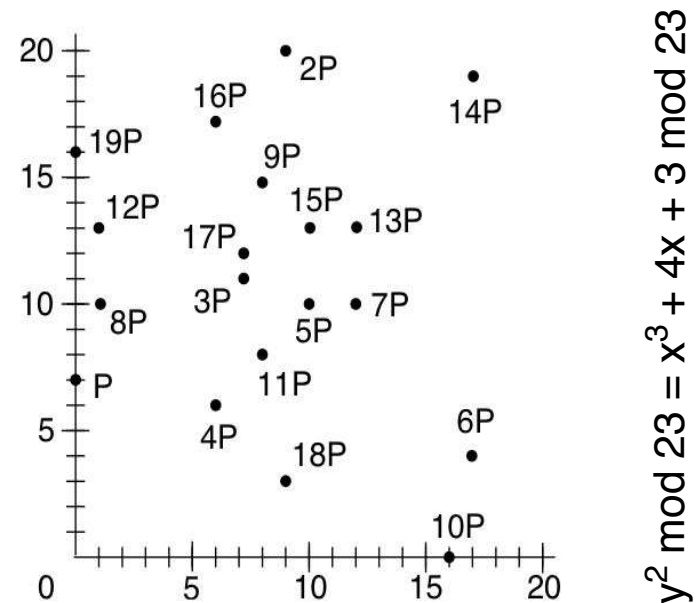
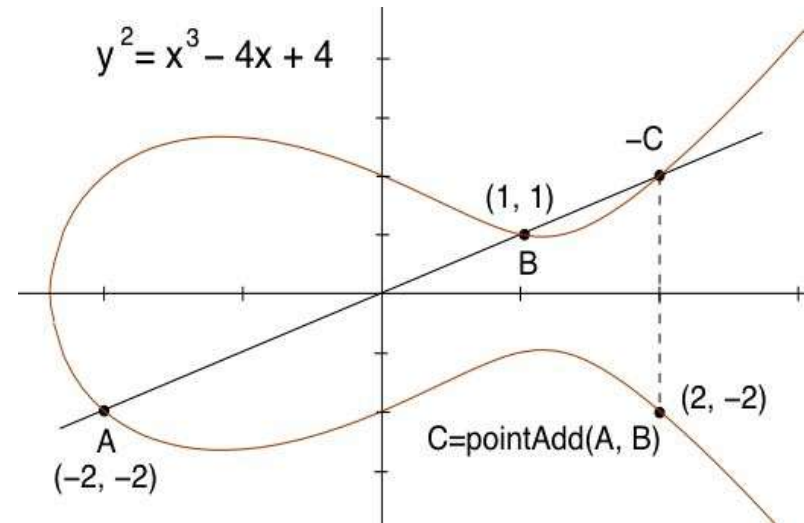
Sym.	RSA	ECC	Ratio	MIPS yrs
80	1,024	160	6:1	10^{12}
112	2,048	224	9:1	10^{24}
128	3,072	256	12:1	10^{28}
192	7,680	384	20:1	10^{47}
256	15,360	521	30:1	10^{66}

Best before 2010

- Computationally highly efficient public-key cryptosystem, highest security strength per bit
 - Savings in memory, bandwidth, power
 - Advantage improves as security needs increase
- Endorsed/standardized by NIST, ANSI, IEEE, IETF
- Good match for AES

How it works

- **Parameters:** Elliptic curve, base point G
- **Scalar point multiplication:** $Q = kP$, e.g. $9P = 2(2(2P)) + P$
- **Hard problem:** Given kP (*public-key*) & P , find k (*private-key*). EC Discrete Logarithm Problem – no known subexponential solutions.



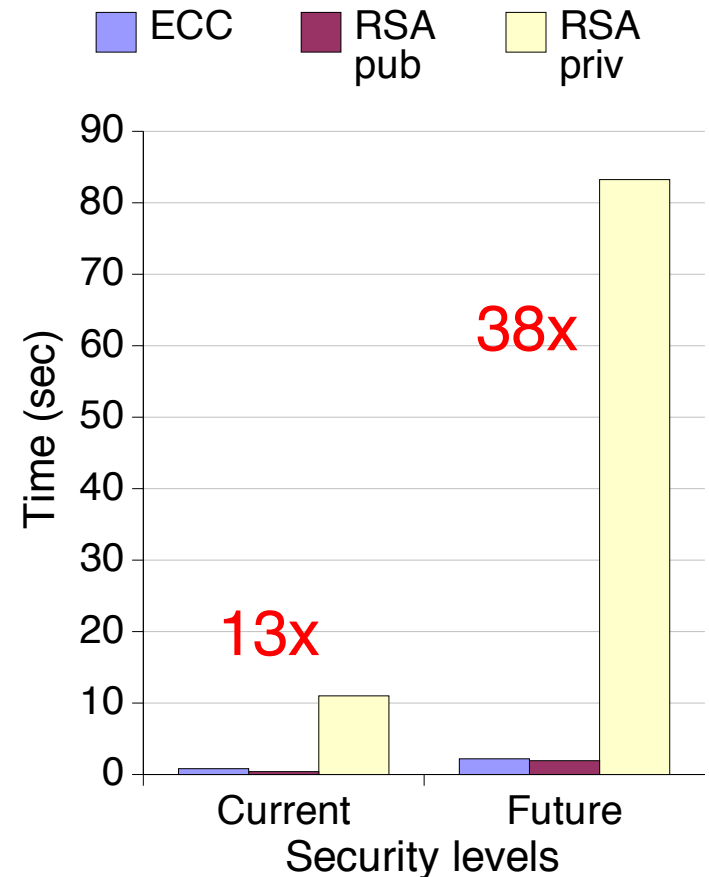
ECC on Small Devices

Berkeley/Crossbow
MICA “mote”
(8-bit, Atmel ATmega
processor, 128KB FLASH,
4KB SRAM, 4KB EEPROM)



Algorithm	Time* (s)	Data bytes	Code bytes
ECC secp160r1	0.81	282	3682
RSA 1024 (priv)	10.99	930	6292
RSA 1024 (pub**)	0.43	542	1073
ECC secp224r1	2.19	422	4812
RSA-2048 (priv)	83.26	1853	7736
RSA-2048 (pub**)	1.94	1332	2854

* 8MHz Atmel ATmega ** e=65537



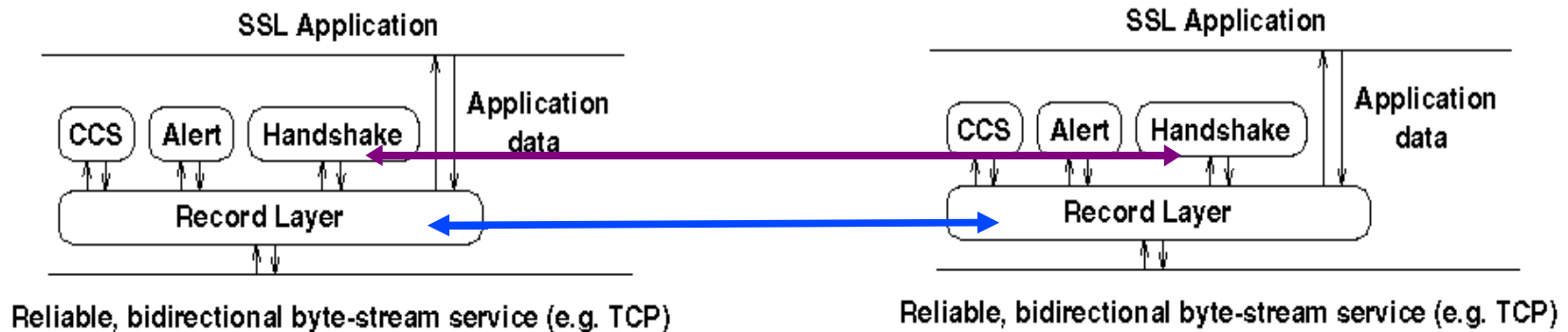
Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- **ECC in Secure Sockets Layer (SSL)**
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Secure Sockets Layer Overview

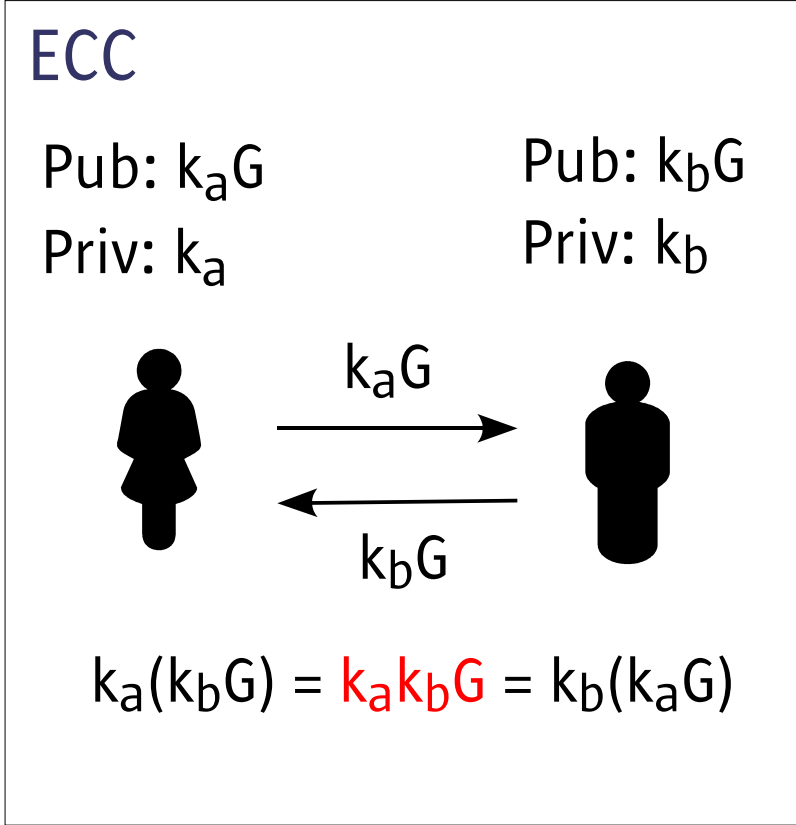
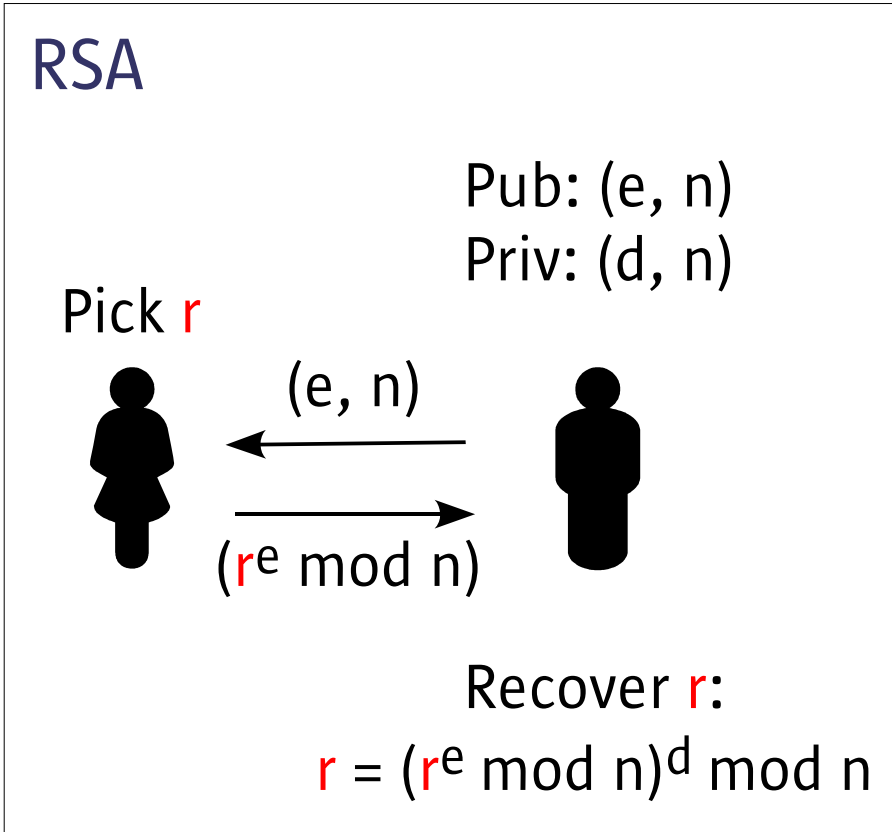
CLIENT

SERVER



- **Handshake protocol** uses expensive public-key operations to establish a **shared secret**
- **Record Layer** uses shared secret to perform (faster) symmetric-key encryption/MAC on application data
- Cipher suite example: RSA_RC4_MD5
- Session reuse amortizes public-key computation across multiple handshakes

Key Exchange



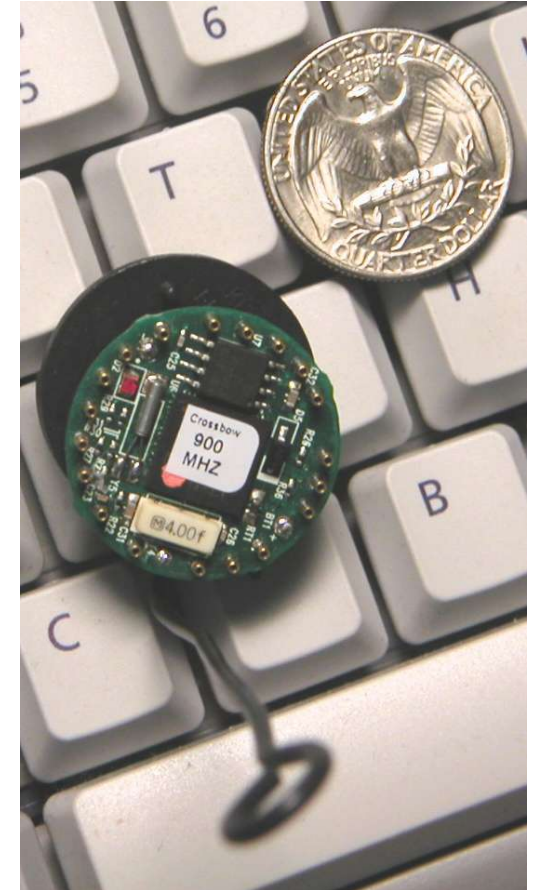
Shared secret can be used to encrypt/decrypt sensitive information

Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Sizzle Overview

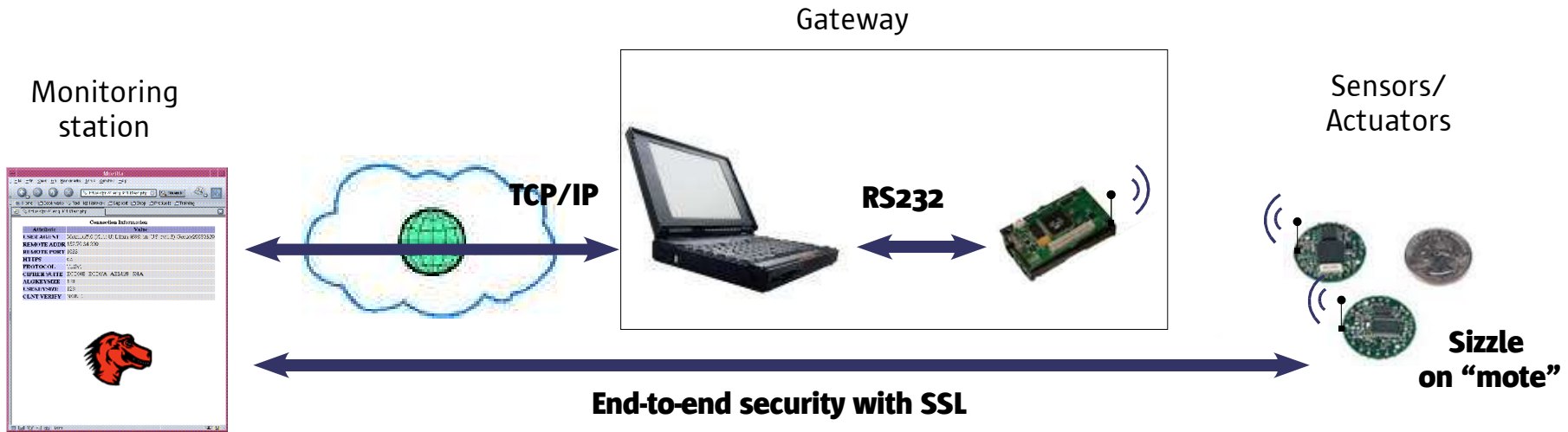
- World's smallest secure web server
- Uses ECC key exchange in SSL*
- Interoperates with ECC-enabled Mozilla/Firefox/OpenSSL
- Lowers barrier for connecting interesting new devices to the Internet, and controlling/monitoring them securely



Sizzle Features

- Uses 160-bit ECC (on curve secp160r1)
- ECDH-ECDSA-RC4-SHA cipher suite
- Minimizes SRAM memory usage and SSL handshake overhead, *e.g.*
 - Static info stored in program memory
 - Small session identifiers, certs
 - Implements session reuse, persistent HTTP(S)

Sizzle Architecture and Statistics

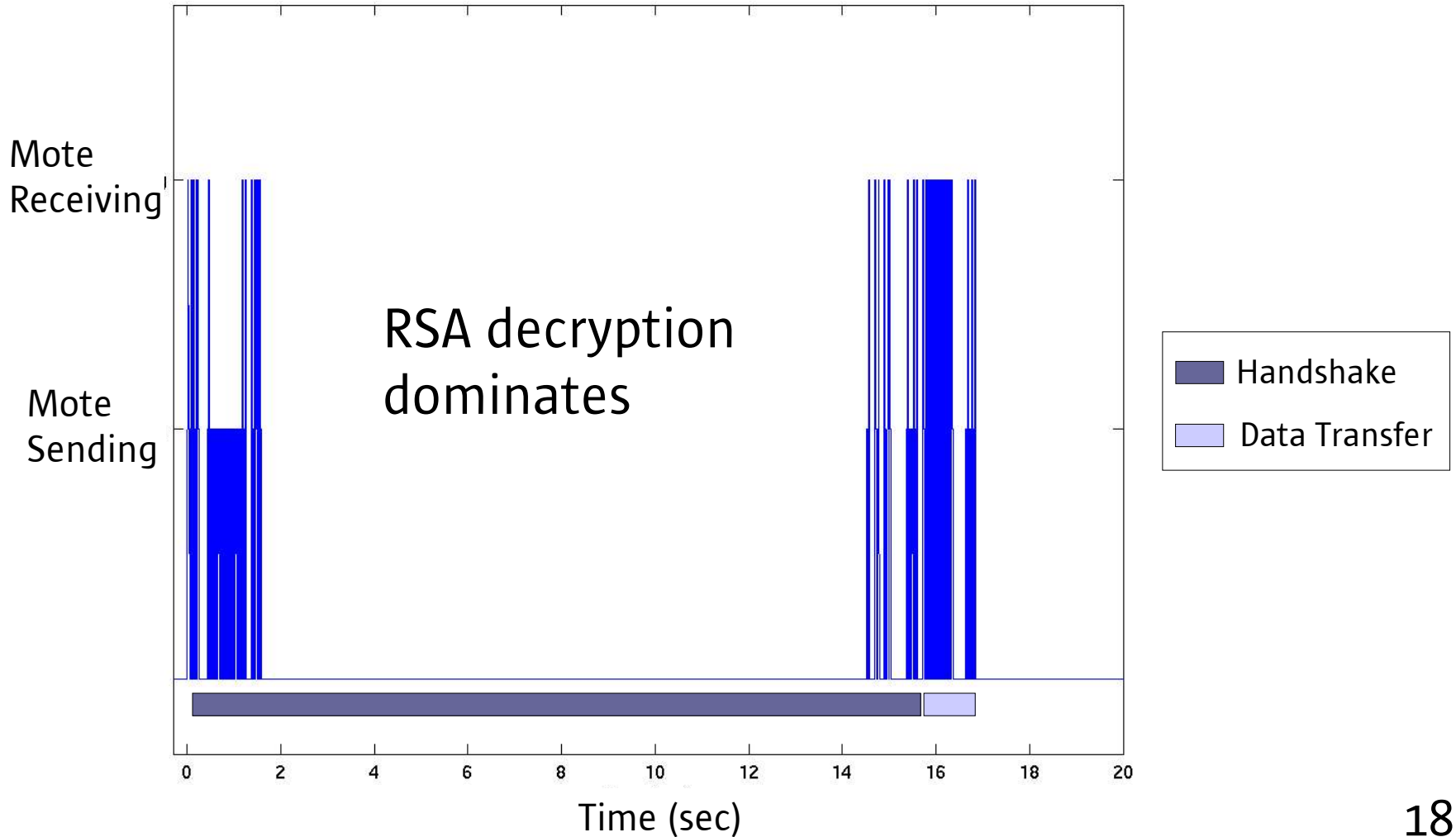


- Mica2 mem usage: ~3KB (RAM), ~60KB (FLASH)
- Page load time in sec (450-byte HTTPS transfer on Mica2 w/ Tiny OS 1.1.6):

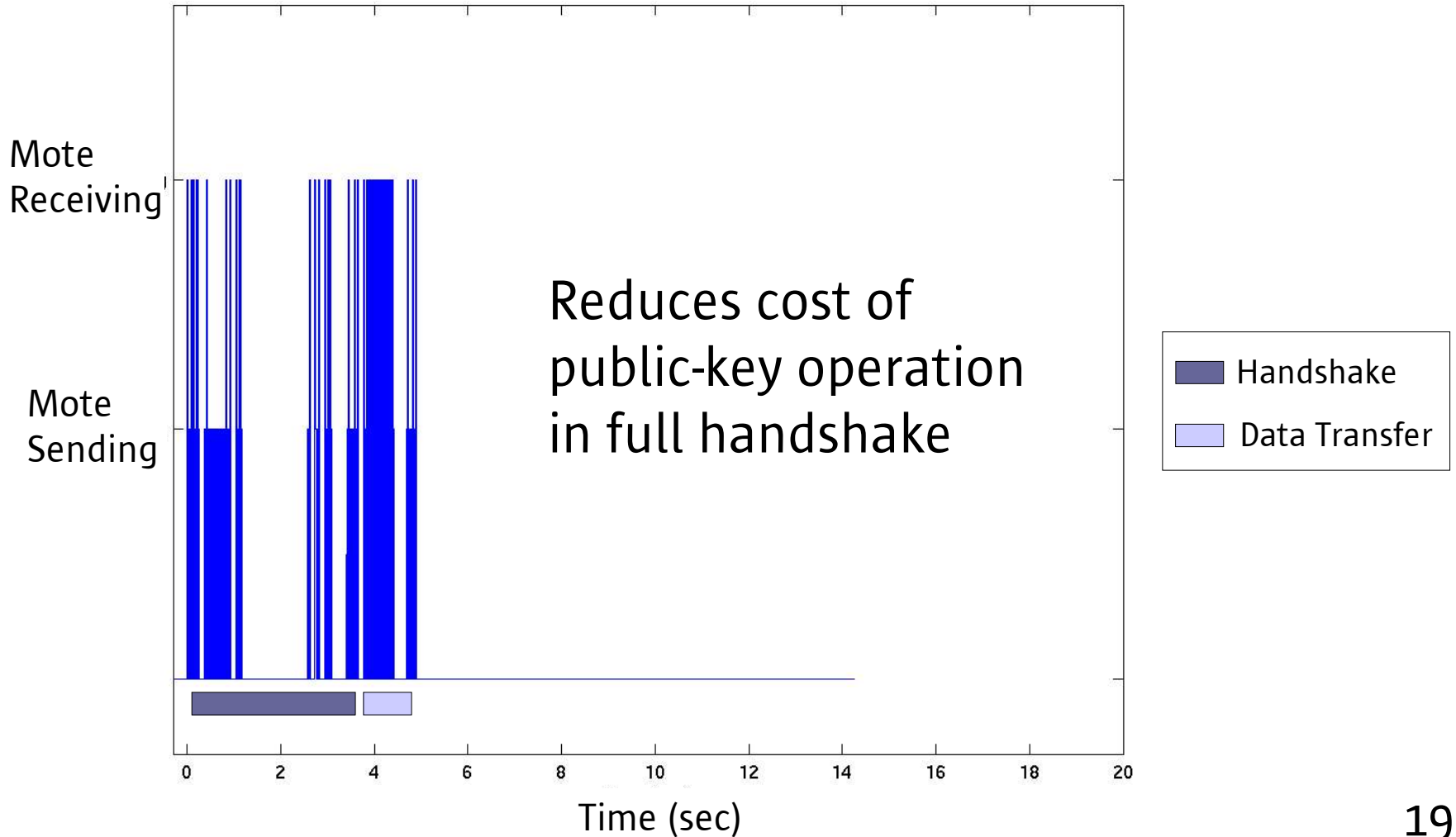
Full Handshake		Session Reuse	Persistent HTTPS*	Plain HTTP*
RSA*	ECC			
16.8	4.9	2.9	1.1	0.9

*New results (subsequent to paper submission)

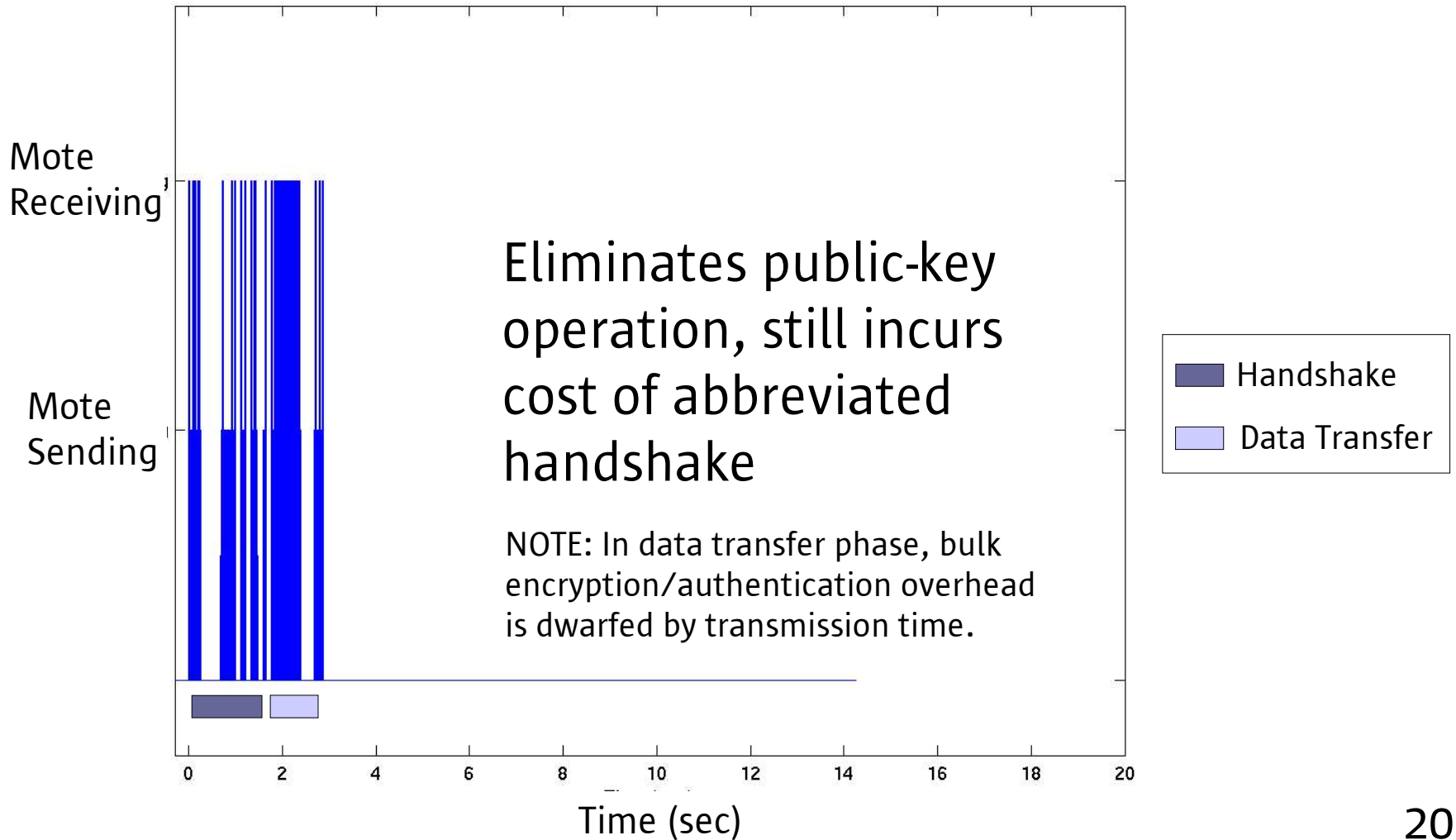
Performance Details (RSA)



Performance Details (ECC)

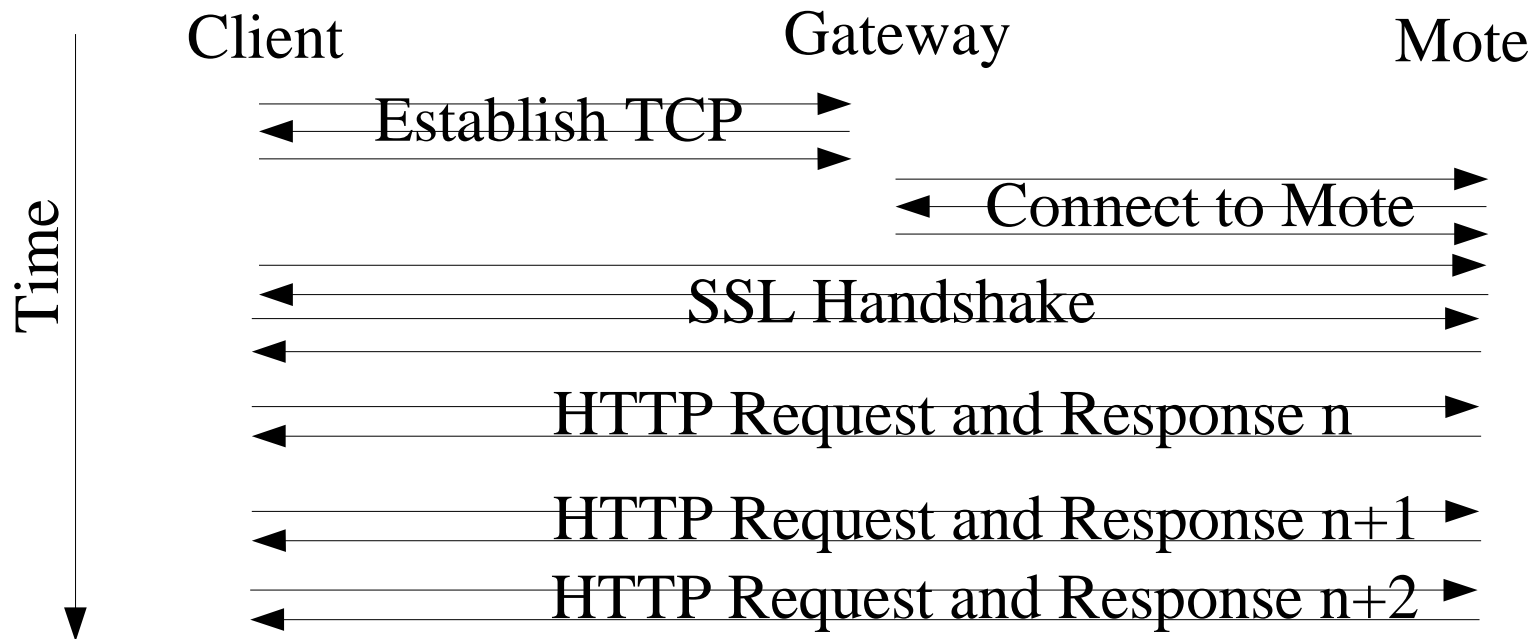


Performance Details (Session Reuse)



Performance Details (Persistent HTTPS)

- Amortizes the cost of an SSL handshake (full or abbreviated) across multiple data transfers



Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Takeaway

Elliptic Curve Cryptography (ECC) makes public-key cryptography feasible on mote-like devices and creates the opportunity to reuse standard security protocols on the “embedded” Internet.

Further Information

- V. Gupta et *al.*, “Sizzle: A Standards-based end-to-end Security Architecture for the Embedded Internet”, PerCom 2005, Mar. 2005
- N. Gura et *al.*, “Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs”, CHES 2004, Aug. 2004
- V. Gupta et *al.*, “ECC Cipher Suites for TLS”, IETF internet-draft, Dec. 2004
- V. Gupta et *al.*, “Integrating Elliptic Curve Cryptography into the Web's Security Infrastructure”, WWW 2004, May 2004

Outline

- Sensor network security background
- Elliptic Curve Cryptography (ECC) overview
- ECC in Secure Sockets Layer (SSL)
- Sizzle (Slim SSL) – HTTPS server on motes
- Conclusion
- Demo

Sizzle Demonstration

- ECC-enabled Mozilla communicating with Sizzle
- Secure monitoring and control of a “wireless thermostat”
- Comparison of ECC v/s RSA-based handshake
- Impact of session reuse and persistent HTTPS



Thank you

sheueling.chang@sun.com

hans.eberle@sun.com

vipul.gupta@sun.com

nils.gura@sun.com

<http://research.sun.com/projects/crypto>

Extra Slides

SSL Handshake: A Closer Look

```

SSL_connect:before/connect initialization
write to 0x609880 [0x186000] (55 bytes => 55 (0x37))
0000 - 80 35 01 03 01 00 0c 00-00 00 20 00 00 48 00 00
0010 - 04 01 00 80 00 00 05 d4-ef be 94 db 4f 4a a0 aa
0020 - cd d2 30 1b 0b 85 41 9f-d1 a0 ac 6f 9e 9a 41 a3
0030 - c1 aa a4 fd e0 c7 01
SSL_connect:SSLv2/v3 write client hello A
read from 0x609880 [0x18c000] (7 bytes => 7 (0x7))
0000 - 16 03 00 01 19 02
0007 - <SPACES/NULS>
read from 0x609880 [0x18c007] (279 bytes => 279 (0x117))
0000 - 00 2a 03 00 e8 72 c3 f2-b3 16 60 de 8b c9 59 02
0010 - b9 10 32 62 cd b9 41 f7-73 76 f5 d3 db b7 a3 d5
0020 - a3 87 79 7f 04 c4 81 9a-03 00 48 00 0b 00 00 e3
0030 - 00 00 e0 00 00 dd 30 81-da 30 81 9a 02 01 06 30
0040 - 09 06 07 2a 86 48 ce 3d-04 01 30 11 31 0f 30 0d
0050 - 06 03 55 04 03 13 06 53-55 4e 57 2d 45 30 1e 17
0060 - 0d 30 34 30 38 30 36 32-31 33 36 30 33 5a 17 0d
0070 - 30 38 30 39 31 34 32 31-33 36 30 33 5a 30 17 31
0080 - 15 30 13 06 03 55 04 03-13 0c 31 31 32 32 33 33
0090 - 34 34 35 35 36 36 30 3e-30 10 06 07 2a 86 48 ce
00a0 - 3d 02 01 06 05 2b 81 04-00 08 03 2a 00 04 ee 11
00b0 - 9d 01 01 4f 26 5a 62 87-f9 e3 a4 fc cc 88 84 4e
00c0 - bc 8b 46 dc be fa 7d b4-8d 0a ac 1f 01 89 8d f3
00d0 - ab 92 d4 b4 e7 f0 30 09-06 07 2a 86 48 ce 3d 04
00e0 - 01 03 30 00 30 2d 02 15-00 ad 70 97 86 11 fb da
00f0 - 60 a2 a5 af ec bc 79 8f-35 7c ad ce ed 02 14 72
0100 - 57 31 af 99 aa 69 1c 63-27 f9 90 8d 2e 23 5f bf
0110 - c8 79 92 0e
0117 - <SPACES/NULS>
SSL_connect:SSLv3 read server hello A
SSL_connect:SSLv3 read server certificate A
SSL_connect:SSLv3 read server done A
write to 0x609880 [0x2809800] (51 bytes => 51 (0x33))
0000 - 16 03 00 00 2e 10 00 00-2a 29 04 6d a0 e0 8d 9b
0010 - 60 8b c6 95 ab 1b 09 50-4a fa 82 81 d6 67 9c b2
0020 - 04 42 66 44 d2 19 bd 50-41 37 77 13 26 50 cc 66
0030 - 1e f0 98
SSL_connect:SSLv3 write client key exchange A
write to 0x609880 [0x2809800] (6 bytes => 6 (0x6))
0000 - 14 03 00 00 01 01
SSL_connect:SSLv3 write change cipher spec A
write to 0x609880 [0x2809800] (65 bytes => 65 (0x41))
0000 - 16 03 00 00 3c 7d dd 48-d3 81 9b ff 74 99 2a 82
0010 - 1f 56 30 7d 34 78 26 8e-b0 76 fd fb 4c aa f3 05
0020 - 65 50 4b bb c5 f0 54 12-8f 0c a5 11 40 7e 65 22
0030 - 37 f0 41 80 9c 2c 81 b8-1d a6 73 d8 0b ab 06 3e
0040 - 4f
SSL_connect:SSLv3 write finished A
SSL_connect:SSLv3 flush data
read from 0x609880 [0x18c000] (5 bytes => 5 (0x5))
0000 - 14 03 00 00 01
read from 0x609880 [0x18c005] (1 bytes => 1 (0x1))
0000 - 01
read from 0x609880 [0x18c000] (5 bytes => 5 (0x5))
0000 - 16 03 00 00 3c
read from 0x609880 [0x18c005] (60 bytes => 60 (0x3C))
0000 - af ca 8a e7 ca 26 f5 44-c1 25 76 96 55 64 56 da
0010 - 2c 58 a6 e1 23 62 00 a2-b6 e6 b7 95 b3 44 a1 e5
0020 - 30 97 9e 0c 7a 39 4d 0c-5f bb 76 ec db f6 bd 02
0030 - 94 ad e6 94 97 67 2e 3d-83 ec 0f df
SSL_connect:SSLv3 read finished A
---
SSL handshake has read 357 bytes and written 177 bytes
---
New, TLSv1/SSLv3, Cipher is ECDH-ECDSA-RC4-SHA
SSL-Session:
    Protocol      : SSLv3
    Cipher       : ECDH-ECDSA-RC4-SHA
    Session-ID:  C4819A03
    Master-Key:  922F0EE1622F4B61B16AA309FD1ECDDF
                1D18AB038BB81473DC115256EE366E87
                CED1240602EF76F77645633C05CF8D9E
    ---

```

Handshake Data Compression

- Large portions of handshake messages are unchanged between different connections to the same mote server and need not be transmitted explicitly
- Can reduce wireless data transmission by 50% for full handshake, 20% for abbreviated handshake
- Requires gateway to parse (but not decrypt) SSL records
- Security is still end-to-end