CERTICOM WHITEPAPER SERIES

ECC and SCADA Key Management

ROB LAMBERT, CERTICOM

August 2010

certicom™

certicom™

Introduction

The setting for this paper is the situation when SCADA data is protected by symmetric key cryptography, and considers the question of how to manage the symmetric key material in the device and the control nodes.

Symmetric key (e.g. AES) systems provide efficient authenticity protection (the fielded devices and the control nodes are ensured that they are communicating with authorized devices) and confidentiality (the communication between the devices is not readable by other unauthorized devices).

There are modes of symmetric-key ciphers that provide combined encryption and authentication or only authentication. These techniques will be described.

However, symmetric key cryptography requires that both sides of the communication be in possession of the secret key.

How should these keys be managed?

Answers to this question will be given as a series of sub-questions and answers.

What are symmetric keys for? MAC-ing

The symmetric keys deployed in for MAC's are are there to provide authenticity, that is they are combined with the messages so that the receiver can be assured that the holder of a secret key produced the message. An example of such a message authentication is NIST's HMAC. Typically these algorithms produce tags of length k. The adversaries probability of successful forgery of such a message is typically 2^{-k}.

Provably secure MAC's have also been developed (e.g. GCM) using the methods of Wegman and Carter.[1] GMAC (a subset of GCM) can be

used to provide authentication only. This does not need to run the cipher (typically AES-128) on every block of plaintext, and thus is very efficient.

AES blocks are typically provided in hardware for very low-powered systems (e.g. Zigbee low-rate radio devices). GCM can typically be added to such HW devices for an additional 30% cost in hardware.

[1] M. Wegman, L. Carter; "New hash functions and their use in authentication and set equality", *Journal of Computer and System Sciences*, 22:265279, 1981.

certicom™

What are symmetric keys for? Encrypting

The second use of symmetric keys is to encrypt data to provide confidentiality. In the past, the cipher used was often the DES, and after the keysize became insufficient, Triple DES (pre-advanced encryption standard selection). TDES is an extension of the older DES standard allowing multiple encryption, but because of this is weaker than it needs to be for the supplied keysize. It is also very slow in firmware.

The AES is NIST's selected replacement for DES and provides 128,192,256 bit keys.

Modes of AES (CCM,GCM) can be used to provide both encryption and authenticity checks at the same time.

How are keys used?

Keys should be used for only one purpose (separation of usage); If not system can often be subverted. Some algorithms give combined duty with one key (e.g. authentication and encryption CCM,GCM). Keys can also be used for separation of control, e.g. one key for monitoring, one for control commands.

There can be a hierarchy of keys (command, control, override, recovery). An important necessity is the provision of backup keys, which are keys to use in the case of compromised original set (more on recovery issues later).

This all assumes that the cryptographic infrastructure is integrated into the SCADA units, and not just used to secure the communication in general.

How are keys transported?

Symmetric keys are needed at both sides of the communication. These keys could be initialized globally, which would be very convenient, but is not recommended since it provides very fragile security.

If the global key is compromised, the key distribution problem is back, but now much worse, since the units are in the field and are being depended upon to be operational.

A second possibility is for the units to be provisioned with individual symmetric keys. This is much less convenient, and forces (in some way, perhaps implicitly) the installer to become a crypto-officer of the system, and requires coordination of keying and installation. There is now also large key exposure at control hubs and large databases of keys.

certicom™

Assymetric Keys

Assymetric (Public) Keys can be very useful to manage symmetric keys.

Assume the provision of (at least) one (private,public) key pair per device, along with the Authority's public key. In addition, the Authority will sign the device's public key. We can also include system update, and other management keys. In this scenario, devices can initialized in house, then installed and configured in field.

1. **Devices provisioned with private/public key pairs, Public portion signed by authority's private key.**
2. **Provisioned devices can be installed in the field.**
3. **Symmetric keys can be negotiated securely via the asymmetric keys.**
4. **Key's managed via assymetric keys (perhaps higher level asymmetric keys).**

Phased cryptography, that is, negotiating symmetric keys in an initial asymmetric stage provides the best features of both symmetric (speed) and asymmetric cryptography (key management).

How should keys be managed?

Excellent advice is given in NIST SP-800-57, recommendations for Key Management, May 2006. Some of the general ideas and recommendations are discussed here. A cryptoperiod is the period of time that a key is allowed to remain active. NIST recommendation depends on circumstance, but the maximum recommendation is 1-3 years. Keys also require protection in deployment. FIPS 140-2 describes the storage of keys at various levels ofsecurity.

Other issues of key management are public-key validity and proof of possession. When systems are designed, action response plans on compromise of keying materials must be designed as well.

NIST's document also considers key management phase and functions, including:

1. **Initialization**
2. **Key establishment**
3. **Operation**
4. **Re-keying**
5. **Post-operation**

Accountability, audit, and survivability are also concerns of key management.

How should keys be managed?

It is wise to consider standardized algorithms for deployment, since they have had more scrutiny.

Algorithms for consideration are RSA, DH/DSA, ECC. The RSA algorithm has public integers e, n, where $n = p \ q$, where p, q and a third integer d are the private key components.

The characteristics of the RSA algorithm are fast encryption/verify but slow signature/decryption and very slow key generation. RSA also lacks a key-negotiation method, relying instead on key transport.

The Diffie-Hellman (DH) algorithm provides key agreement with modular integer calculations. DH keys are efficiently calculated as $A = g^a \bmod p$, a private, A public, (g, p) are system parameters. The same style of keys can be used in signatures with the DSA algorithm.

Similarly, ECC methods, ECDH and ECDSA are provide for key agreement and signatures. ECC keys are usually written additively, $A = aG$, and now A and G are points on an elliptic curve.

What size should the asymmetric keys be?

To determine the key sizes required for the asymmetric algorithms, the underlying hard problems must be considered. For RSA the hard problem e-th roots modulo n of unknown factorization. This problem is not harder than the problem of integer factorization. For DH the problem is the integer DH problem, which is not harder than integer discrete logarithms. For ECC, the hard problems are not harder than the elliptic curve discrete logarithm problem.

Currently there are subexponential algorithm for integer problems, all based on the number field sieve. There are no subexponential algorithm for cryptographic EC instances, but only Pollard's rho algorithm (and parallelized variants), which is exponential time.

What size should the asymmetric keys be?

To determine the key sizes required for the asymmetric algorithms, the underlying hard problems must be considered. For RSA the hard problem e-th roots modulo n of unknown factorization. This problem is not harder than the problem of integer factorization. For DH the problem is the integer DH problem, which is not harder than integer discrete logarithms. For ECC, the hard problems are not harder than the elliptic curve discrete logarithm problem.