# CWE Detail – CWE-327

## Description

The product uses a broken or risky cryptographic algorithm or protocol.

## Extended Description

Cryptographic algorithms are the methods by which data is scrambled to prevent observation or influence by unauthorized actors. Insecure cryptography can be exploited to expose sensitive information, modify data in unexpected ways, spoof identities of other users or devices, or other impacts. It is very difficult to produce a secure algorithm, and even high-profile algorithms by accomplished cryptographic experts have been broken. Well-known techniques exist to break or weaken various kinds of cryptography. Accordingly, there are a small number of well-understood and heavily studied algorithms that should be used by most products. Using a non-standard or known-insecure algorithm is dangerous because a determined adversary may be able to break the algorithm and compromise whatever data has been protected. Since the state of cryptography advances so rapidly, it is common for an algorithm to be considered "unsafe" even if it was once thought to be strong. This can happen when new attacks are discovered, or if computing power increases so much that the cryptographic algorithm no longer provides the amount of protection that was originally thought. For a number of reasons, this weakness is even more challenging to manage with hardware deployment of cryptographic algorithms as opposed to software implementation. First, if a flaw is discovered with hardware-implemented cryptography, the flaw cannot be fixed in most cases without a recall of the product, because hardware is not easily replaceable like software. Second, because the hardware product is expected to work for years, the adversary's computing power will only increase over time.

## Threat-Mapped Scoring

Score: 3.0

Priority: P2 - Serious (High)

## Observed Examples (CVEs)

**•** CVE-2022-30273: SCADA-based protocol supports a legacy encryption mode that uses Tiny Encryption Algorithm (TEA) in ECB mode, which leaks patterns in messages and cannot protect integrity

**•** CVE-2022-30320: Programmable Logic Controller (PLC) uses a protocol with a cryptographically insecure hashing algorithm for passwords.

**•** CVE-2008-3775: Product uses "ROT-25" to obfuscate the password in the registry.

**•** CVE-2007-4150: product only uses "XOR" to obfuscate sensitive data

**•** CVE-2007-5460: product only uses "XOR" and a fixed key to obfuscate sensitive data

**•** CVE-2005-4860: Product substitutes characters with other characters in a fixed way, and also leaves certain input characters unchanged.

**•** CVE-2002-2058: Attackers can infer private IP addresses by dividing each octet by the MD5 hash of '20'.

**•** CVE-2008-3188: Product uses DES when MD5 has been specified in the configuration, resulting in weaker-than-expected password hashes.

**•** CVE-2005-2946: Default configuration of product uses MD5 instead of stronger algorithms that are available, simplifying forgery of certificates.

**•** CVE-2007-6013: Product uses the hash of a hash for authentication, allowing attackers to gain privileges if they can obtain the original hash.

## Related Attack Patterns (CAPEC)

* CAPEC-20
* CAPEC-459
* CAPEC-473
* CAPEC-475
* CAPEC-608
* CAPEC-614
* CAPEC-97

## Attack TTPs

**•** T1553.002: Code Signing (Tactics: defense-evasion)

**•** T1036.001: Invalid Code Signature (Tactics: defense-evasion)

## Modes of Introduction

**•** Architecture and Design: COMMISSION: This weakness refers to an incorrect design related to an architectural security tactic.

**•** Implementation: With hardware, the Architecture or Design Phase might start with compliant cryptography, but it is replaced with a non-compliant crypto during the later Implementation phase due to implementation constraints (e.g., not enough entropy to make it function properly, or not enough silicon real estate available to implement). Or, in rare cases (especially for long projects that span over years), the Architecture specifications might start with cryptography that was originally compliant at the time the Architectural specs were written, but over the time it became non-compliant due to progress made in attacking the crypto.

## Common Consequences

**•** Impact: Read Application Data — Notes: The confidentiality of sensitive data may be compromised by the use of a broken or risky cryptographic algorithm.

**•** Impact: Modify Application Data — Notes: The integrity of sensitive data may be compromised by the use of a broken or risky cryptographic algorithm.

**•** Impact: Hide Activities — Notes: If the cryptographic algorithm is used to ensure the identity of the source of the data (such as digital signatures), then a broken algorithm will compromise this scheme and the source of the data cannot be proven.

## Potential Mitigations

**•** Architecture and Design: When there is a need to store or transmit sensitive data, use strong, up-to-date cryptographic algorithms to encrypt that data. Select a well-vetted algorithm that is currently considered to be strong by experts in the field, and use well-tested implementations. As with all cryptographic mechanisms, the source code should be available for analysis. For example, US government systems require FIPS 140-2 certification [REF-1192]. Do not develop custom or private cryptographic algorithms. They will likely be exposed to attacks that are well-understood by cryptographers. Reverse engineering techniques are mature. If the algorithm can be compromised if attackers find out how it works, then it is especially weak. Periodically ensure that the cryptography has not become obsolete. Some older algorithms, once thought to require a billion years of computing time, can now be broken in days or hours. This includes MD4, MD5, SHA1, DES, and other algorithms that were once regarded as strong. [REF-267] (Effectiveness: N/A)

**•** Architecture and Design: Ensure that the design allows one cryptographic algorithm to be replaced with another in the next generation or version. Where possible, use wrappers to make the interfaces uniform. This will make it easier to upgrade to stronger algorithms. With hardware, design the product at the Intellectual Property (IP) level so that one cryptographic algorithm can be replaced with another in the next generation of the hardware product. (Effectiveness: Defense in Depth)

**•** Architecture and Design: Carefully manage and protect cryptographic keys (see CWE-320). If the keys can be guessed or stolen, then the strength of the cryptography itself is irrelevant. (Effectiveness: N/A)

**•** Architecture and Design: Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid. Industry-standard implementations will save development time and may be more likely to avoid errors that can occur during implementation of cryptographic algorithms. Consider the ESAPI Encryption feature. (Effectiveness: N/A)

**•** Implementation: When using industry-approved techniques, use them correctly. Don't cut corners by skipping resource-intensive steps (CWE-325). These steps are often essential for preventing common attacks. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

**•** Verilog (Class: None, Prevalence: Undetermined)

**•** VHDL (Class: None, Prevalence: Undetermined)

## Demonstrative Examples

**•** Once considered a strong algorithm, DES now regarded as insufficient for many applications. It has been replaced by Advanced Encryption Standard (AES).

**•** However, SHA1 was theoretically broken in 2005 and practically broken in 2017 at a cost of $110K. This means an attacker with access to cloud-rented computing power will now be able to provide a malicious bitstream with the same hash value, thereby defeating the purpose for which the hash was used.

**•** Multiple OT products used weak cryptography.

## Notes

**•** Maintenance: Since CWE 4.4, various cryptography-related entries, including CWE-327 and CWE-1240, have been slated for extensive research, analysis, and community consultation to define consistent terminology, improve relationships, and reduce overlap or duplication. As of CWE 4.6, this work is still ongoing.

**•** Maintenance: The Taxonomy\_Mappings to ISA/IEC 62443 were added in CWE 4.10, but they are still under review and might change in future CWE versions. These draft mappings were performed by members of the "Mapping CWE to 62443" subgroup of the CWE-CAPEC ICS/OT Special Interest Group (SIG), and their work is incomplete as of CWE 4.10. The mappings are included to facilitate discussion and review by the broader ICS/OT community, and they are likely to change in future CWE versions.