A Counterspherics
Think Tank Mini Paper

VENUM: A Novel & Extensible Data Security Model Supporting Variable Patterns for SPI/PII

By R. Prince for Counterspherics Labs

Editors & Contributors:

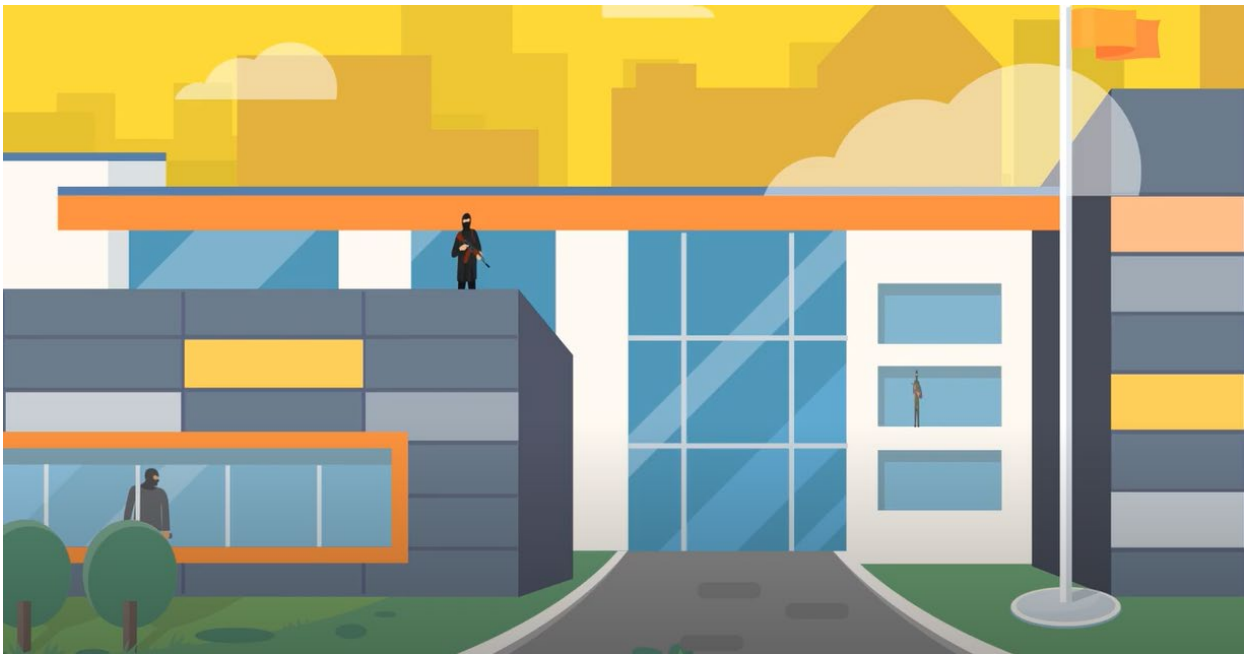
- C. Meiers,
  Counterspherics Think Tank Member
- K. Pearson
  Counterspherics Think Tank Member

Counterspherics Inc.

Technology for Operating at the Speed of Now!

Counterspherics…engineers of MERLENN™, a Real-time, Secure, Contingency (emergency) Management & Support System in preparation for, responding to, & recovering from, Human, Atmospheric, Geological, Technological, and Biological Threats.

The past is prologue. Your Business is Ft. Knox, and your information is the Gold...



The Challenge

I am a customer of Target. But I no longer enter my PIN anywhere. Once it was announced that my data at Target had potentially fallen into nefarious hands (well after the incident occurred, of course), it did not matter "how" or "why" the breach occurred. My shopping experience had changed forever. I use a restricted account for online and bricks/mortar retail transactions. Target changed the way I do business with the entire industry as well as causing me to question how well they are protecting my SPI/PII. I am not the only customer thinking this way.

As news channels, security experts, newspaper articles, websites, and books shine a bright spotlight on dramatically increasing numbers of cases where data security all went wrong, customers are holding businesses accountable for protection of their SPI/PII (Sensitive Personal Information/Personal Identifying Information).

Target, Experian, Equifax, Ebay, Capital One, Marriott, Linked-In, and Sony - to name a few – have suffered embarrassing and damaging examples of data security failures that all business leaders should be aware. The time for serious attention to data and application security is paramount and forefront and should not be the afterthought it has been.

Security Architecture

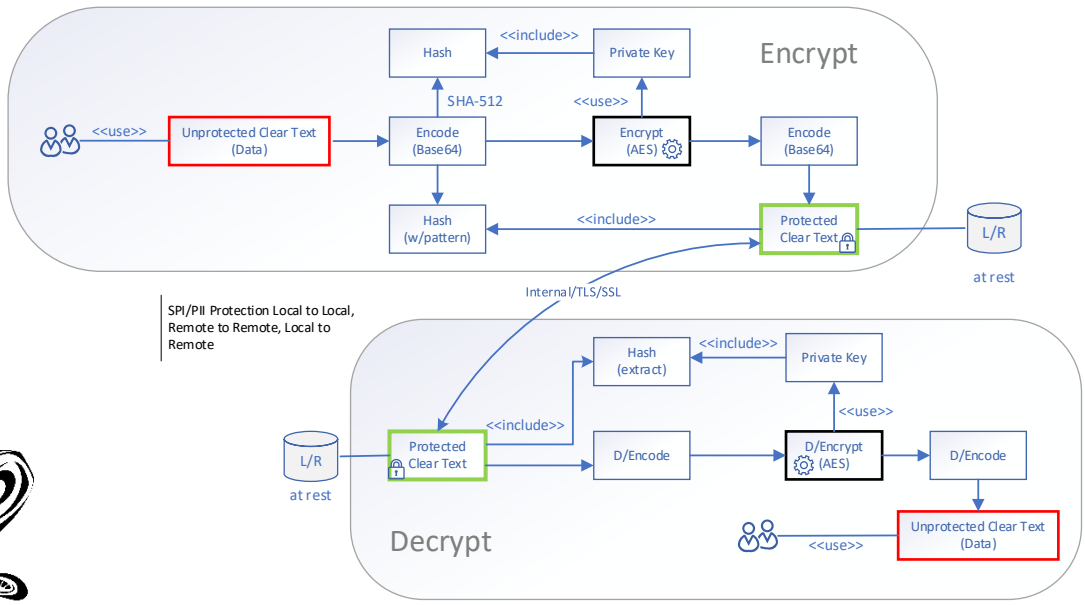
Businesses hire analysts, designers, architects, and engineers who have backgrounds in theoretical and applied security models, frameworks, and information encryption to create and maintain their operational security environment. These technology professionals are the target audience for this brief paper. Herein I introduce a novel digital security framework that has worked for me and remains stout over time.

Not many things made by humans are invincible, unbreakable, impregnable, or impervious. We assert that there are new ways and treatments of thinking to create a useable, repeatable, data security outcome. Like changing your password on a regular basis, VENUM is taxonomy of a class of extensible security patterns which can be applied to web-based as well as client-side applications. A pattern that is "living". One that allows for it to dynamically evolve as security needs change over time. A decoupled security architecture which applies simple but effective varied patterns of encryption is a novel way of accomplishing the goal. Data is input and encrypted providing an output either from a function within an application or from a larger system/service that can choose which pattern(s) to apply data.

What does this look like?

COUNTERSPHERICS

Variable Encryption Narrowcast Unimodality Model
Using DEHSE Ecrypting Pattern
(VENUM)



MERLENN

VENUM…

Introducing VENUM (the "Varied Encryption Narrowcast Unimodality Model") that supports multiple types of security encryption and patterns. VENUM is both a data security architecture supporting varying data security patterns of encryption. VENUM is an extensible, no-frills, and light data security model that is easy to implement. VENUM is a new realization of an older model. If you are someone who is coding, encrypting, and converting the resultant data to a string for storage or transmission…is routine. Nothing new here. However, the VENUM taxonomy is a new treatment of a data security approach that further secures the data with little effort.

The model above depicts a DEHSE or Double Encoding Hashed Single Encryption pattern. In VENUM there are many variations of the VENUM patterns e.g.

- Double Encoded Hashed Double Encryption of the SPI/PII
- Single Encoding Hashed Single Encryption of the SPI/PII
- Single Encoding Hashed Double Encryption of the SPI/PII

…or any variant thereof. The constant is the Hash. The Hash is central to VENUM. I'll explain…

Turn On The Lights!

To sum it up, there is a novel elegance to this. The treatment of the Hash defines the VENUM class of patterns. If it has not hit you yet***, the Hash is both the venom that "poisons" the final encoded encrypted output and the extracted "antivenin" which restores it***. Without knowing the Hash's fractional segments and pattern i.e. spread and size(s) of the lengths in the final encoded encrypted output, the data cannot be returned to its original state. To decrypt with VENUM you simply do an "about face" and "run backwards."

In the model above you have the unprotected clear/plain text SPI/PII data in its normal form. The flow to protect (encrypt) as shown is:

1. Extract the Hash from the encoded encrypted data
2. Reassemble the encoded Hash in its sequential parts
3. Unencode the encoded encrypted data
4. Recombine the Hash with the secret key
5. Decrypt the data using the secret key
6. Unencode the decrypted data again (convert back to text)
7. Unencoded Unencrypted data is now present in its source state

The beauty of this is…follow me here…the encoded encrypted data is again changed by interspersing the Hash throughout its textual body. Transformed. This essentially encrypts the

data without running it through the encrypting process a second time. In essence one can say that technically this is symmetric encryption and logically it behaves like asymmetric encryption if you think of the interspersed Hash as a public key. This is a conversation for another day and I know stirs the pot! The result is that the interspersed Hash accomplishes two things.

1. It serves to make the resulting output harder to decrypt and stores itself within the encrypted text as text for later recall. This is efficient. The process for storing the Hash in parts as interspersed elements throughout the encrypted data as part of its textual representation makes it indistinguishable from the rest of the whole.
2. As text it is easily transmitted and stored and resource friendly. Only the generating process knows the locations (pattern) of the interspersed Hash. Recombination of the Hash from the encrypted textual representation allows for its decryption thus no separate storage mechanism for the Hash is needed – that is, unless you want or this need arises.

The real fun and novelty begins with creating pattern variants and pattern extensibility. One can encrypt the Hash in a separate process, use a storage mechanism to persist it, then like the data, encode it before interspersing it with the encrypted encoded data output. In this variant we are encrypting and encoding the Hash just as we did the data. When you really think about it…we are using the Hash (unencrypted or encrypted but always encoded) as a form of additional encryption methodology and key(s). Why do this? You might want two ways to add "venom" to your result…one with an Encoded Encrypted Hash and one without.

It's all open and based on your innovative thinking and needs. Without the Hash there is no VENUM. In the VENUM DESE pattern Double Encoding is the encoding of the Hash and the encoding the Encrypted result***. In reality you are not relegated to an interspersed Hash…rather any encoded or randomized string ot other text that is spread throughout the body of the final result does the trick. Remember that in this pattern the string is also part of the encryption key. It does not have to be. In any case the impact of the "poisoning" of the final result is still a realization of the VENUM Model***.

Lastly, remember this model can be implemented as a decoupled component. Security capability can be increased without changing your architectural structure. Like changing passwords, with VENUM you have an active security model that allows the application of on demand varied encryption types, patterns, and venom, to secure SPI/PII multiple ways.

The above model as implemented in ASP.NET is around 34 lines of source (just encryption) and uses the System.Security.Cryptography namespace to implement AES. The Hash is SHA-512 and generated in a co-working process.

About the Author: *Mr. Prince is a Chief Technologist and part of the strategic think tank at Counterspherics Labs. He is a consultant with over 20 years of IT/IS Strategic and Operational Alignment, Analysis, Design, Architecture and Engineering, experience. He lives in Southern California, with his wife. They "three girls and a boy."*

DON'T GET BIT. BYTE BACK!

COUNTERSPHERICS

Distributed By

Counterspherics Inc.
415 W Foothill Blvd Suite 233
Claremont, CA 91711

Phone (877) 205-1124 Ext. 1537
Learn more at www.counterspherics.com