PURB (cryptography)

In cryptography, a **padded uniform random blob** or **PURB** is a discipline for encrypted data formats designed to minimize unintended information leakage either from its encryption format metadata or from its total length.[1]

Properties of PURBs

When properly created, a PURB's content is indistinguishable from a uniform random bit string to any observer without a relevant decryption key. A PURB therefore leaks *no* information through headers or other cleartext metadata associated with the encrypted data format. This leakage minimization "hygiene" practice contrasts with traditional encrypted data formats such as Pretty Good Privacy, which include cleartext metadata encoding information such as the application that created the data, the data format version, the number of recipients the data is encrypted for, the identities or public keys of the recipients, and the ciphers or suites that were used to encrypt the data. While such encryption metadata was considered non-sensitive when these encrypted formats were designed, modern attack techniques have found numerous ways to employ such incidentally-leaked metadata in facilitating attacks, such as by identifying data encrypted with weak ciphers or obsolete algorithms, fingerprinting applications to track users or identify software versions with known vulnerabilities, or traffic analysis techniques such as identifying all users, groups, and associated public keys involved in a conversation from an encrypted message observed between only two of them.

In addition, a PURB is padded to a constrained set of possible lengths, in order to minimize the amount of information the encrypted data could potentially leak to observers via its total length. Without padding, encrypted objects such as files or bit strings up to M bits in length can leak up to $O(\log M)$ bits of information to an observer - namely the number of bits required to represent the length exactly. A PURB is padded to a length representable in a floating point number whose mantissa is no longer (i.e., contains no more significant bits) than its exponent. This constraint limits the maximum amount of information a PURB's total length can leak to $O(\log \log M)$ bits, a significant asymptotic reduction and the best achievable in general for variable-length encrypted formats whose multiplicative overhead is limited to a constant factor of the unpadded payload size. This asymptotic leakage is the same as one would obtain by padding encrypted objects to a power of some base, such as to a power of two. Allowing some significant mantissa bits in the length's representation rather than just an exponent, however, significantly reduces the overhead of padding. For example, padding to the next power of two can impose up to 100% overhead by nearly doubling the object's size, while a PURB's padding imposes overhead of at most 12% for small strings and decreasing gradually (to 6%, 3%, etc.) as objects get larger.

Experimental evidence indicate that on data sets comprising objects such as files, software packages, and online videos, leaving objects unpadded or padding to a constant block size often leaves them uniquely identifiable by total length alone.[2][3][1] Padding objects to a power of two or to a PURB length, in contrast, ensures that most objects are indistinguishable from at least some other objects and thus have a nontrivial *anonymity set*.[1]

Encoding and decoding PURBs

Because a PURB is a discipline for designing encrypted formats and not a particular encrypted format, there is no single prescribed method for encoding or decoding PURBs. Applications may use any encryption and encoding scheme provided it produces a bit string that appears uniformly random to an observer without an appropriate key, provided the appropriate hardness assumptions are satisfied of course, and provided the PURB is padded to one of the allowed lengths. Correctly-encoded PURBs therefore *do not identify the application that created them* in their ciphertext. A decoding application, therefore, cannot readily tell before decryption whether a PURB was encrypted for that application or its user, other than by trying to decrypt it with any available decryption keys.

Encoding and decoding a PURB presents technical efficiency challenges, in that traditional parsing techniques are not applicable because a PURB by definition has no metadata *markers* that a traditional parser could use to discern the PURB's structure before decrypting it. Instead, a PURB must be *decrypted first* obliviously to its internal structure, and then parsed only after the decoder has used an appropriate decryption key to find a suitable cryptographic *entrypoint* into the PURB.

Encoding and decoding PURBs intended to be decrypted by several different recipients, public keys, and/or ciphers presents the additional technical challenge that each recipient must find a different entrypoint at a distinct location in the PURB non-overlapping with those of the other recipients, but the PURB presents no cleartext metadata indicating the positions of those entrypoints or even the total number of them. The paper that proposed PURBs[1] also included algorithms for encrypting objects to multiple recipients using multiple cipher suites. With these algorithms, recipients can find their respective entrypoints into the PURB with only a logarithmic number of *trial decryptions* using symmetric-key cryptography and only one expensive public-key operation per cipher suite.

A third technical challenge is representing the public-key cryptographic material that needs to be encoded into each entrypoint in a PURB, such as the ephemeral Diffie-Hellman public key a recipient needs to derive the shared secret, in an encoding indistinguishable from uniformly random bits. Because the standard encodings of elliptic-curve points are readily distinguishable from random bits, for example, special *indistinguishable* encoding algorithms must be used for this purpose, such as Elligator[4] and its successors.[5][6]

Tradeoffs and limitations

The primary privacy advantage that PURBs offer is a strong assurance that correctly-encrypted data leaks nothing incidental via internal metadata that observers might readily use to identify weaknesses in the data or software used to produce it, or to fingerprint the application or user that

created the PURB. This privacy advantage can translate into a security benefit for data encrypted with weak or obsolete ciphers, or by software with known vulnerabilities that an attacker might exploit based on trivially-observable information gleaned from cleartext metadata.

A primary disadvantage of the PURB encryption discipline is the complexity of encoding and decoding, because the decoder cannot rely on conventional parsing techniques before decryption. A secondary disadvantage is the overhead that padding adds, although the padding scheme proposed for PURBs incurs at most only a few percent overhead for objects of significant size.

The Padme padding proposed in the PURB paper only creates files of specific very distinct sizes. Thus, an encrypted file may often be identified as PURB encrypted with high confidence, as the probability of any other file having exactly one of those padded sizes is very low. Another padding problem occurs with very short messages, where the padding does not effectively hide the size of the content.

One critique of incurring the complexity and overhead costs of PURB encryption is that the *context* in which a PURB is stored or transmitted may often leak metadata about the encrypted content anyway, and such metadata is outside of the encryption format's purview or control and thus cannot be addressed by the encryption format alone. For example, an application's or user's choice of filename and directory in which to store a PURB on disk may indicate allow an observer to infer the application that likely created it and to what purpose, even if the PURB's data content itself does not. Similarly, encrypting an E-mail's body as a PURB instead of with traditional PGP or S/MIME format may eliminate the encryption format's metadata leakage, but cannot prevent information leakage from the cleartext E-mail headers, or from the endpoint hosts and E-mail servers involved in the exchange. Nevertheless, separate but complementary disciplines are typically available to limit such contextual metadata leakage, such as appropriate file naming conventions or use of pseudonymous E-mail addresses for sensitive communications.

References

1. Nikitin, Kirill; Barman, Ludovic; Lueks, Wouter; Underwood, Matthew; Hubaux, Jean-Pierre; Ford, Bryan (2019). "Reducing Metadata Leakage from Encrypted Files and Communication with PURBs" (https://petsymposium.org/2019/files/papers/issue4/popets-2019-0056.pdf) (PDF). *Proceedings on Privacy Enhancing Technologies (PoPETS)*. **2019** (4): 6–33. arXiv:1806.03160 (https://arxiv.org/abs/1806.03160). doi:10.2478/popets-2019-0056 (https://doi.org/10.2478%2Fpopets-2019-0056). S2CID 47011059 (https://api.semanticscholar.org/CorpusID:47011059).
2. Hintz, Andrew (April 2002). *Fingerprinting Websites Using Traffic Analysis*. International Workshop on Privacy Enhancing Technologies. doi:10.1007/3-540-36467-6_13 (https://doi.org/10.1007%2F3-540-36467-6_13).
3. Sun, Qixiang; Simon, D.R.; Wang, Yi-Min; Russell, W.; Padmanabhan, V.N.; Qiu, Lili (May 2002). *Statistical Identification of Encrypted Web Browsing Traffic*. IEEE Symposium on Security and Privacy. doi:10.1109/SECPRI.2002.1004359 (https://doi.org/10.1109%2FSECPRI.2002.1004359).
4. Bernstein, Daniel J.; Hamburg, Mike; Krasnova, Anna; Lange, Tanja (November 2013). *Elligator: Elliptic-curve points indistinguishable from uniform random strings* (https://dl.acm.org/citation.cfm?id=2516734). Computer Communications Security (https://dl.acm.org/citation.cfm?id=2508859).

5. Tibouchi, Mehdi (March 2014). *Elligator Squared: Uniform Points on Elliptic Curves of Prime Order as Uniform Random Strings* (https://www.ifca.ai/fc14/papers/fc14_submission_25.pdf) (PDF). Financial Cryptography and Data Security (https://www.ifca.ai/fc14/index.html).
6. Aranha, Diego F.; Fouque, Pierre-Alain; Qian, Chen; Tibouchi, Mehdi; Zapalowicz, Jean-Christophe (August 2014). *Binary Elligator Squared* (https://eprint.iacr.org/2014/486.pdf) (PDF). International Conference on Selected Areas in Cryptography.
