Abstract

Encoding is often discussed alongside security, but it is not security by itself. This whitepaper explains how Base64 and Bech32 work, why they are used, where they are efficient, and where they are commonly misunderstood. Base64 is a general-purpose binary-to-text encoding defined by RFC 4648. Bech32 is a Bitcoin address format specified by BIP-173 that prioritizes readability and error detection. The paper also distinguishes encoding from encryption, hashing, and Secure Multi-Party Computation (MPC), avoiding misleading comparisons between unrelated technical layers.

Base64 overhead

~33%

3 bytes become 4 characters

Padding

small

padding is not the main cost

Bech32 checksum

built-in

detects many human errors

Encoding security

none

not encryption or hashing

Executive summary

Topic	Correct interpretation
Base64	Best for moving binary data through text-only systems such as APIs, email, JSON fields, and tokens. Its main cost is the 3-byte to 4-character expansion, not the final padding alone.
Bech32	Best for Bitcoin-style address strings where human readability and error detection matter more than general-purpose binary transport.
Security boundary	Encoding does not hide data. Use encryption for confidentiality, hashes or MACs for integrity, and digital signatures for authenticity.
MPC distinction	MPC is a cryptographic computation model. It should be discussed as a separate privacy technology, not as a direct alternative to Bech32 or Base64.

1. Why Encoding Matters

Network protocols, web APIs, email systems, and blockchains often need to represent binary data using safe printable characters. Encoding handles that representation problem. A correct protocol design chooses an encoding format based on transport constraints, readability, validation needs, and compatibility with existing standards.

A common mistake is to treat encoding as a security mechanism. Encoded data can usually be decoded without a secret key. Therefore, an encoded password, token payload, or address is not automatically protected. Security must come from cryptographic controls around the encoded data.

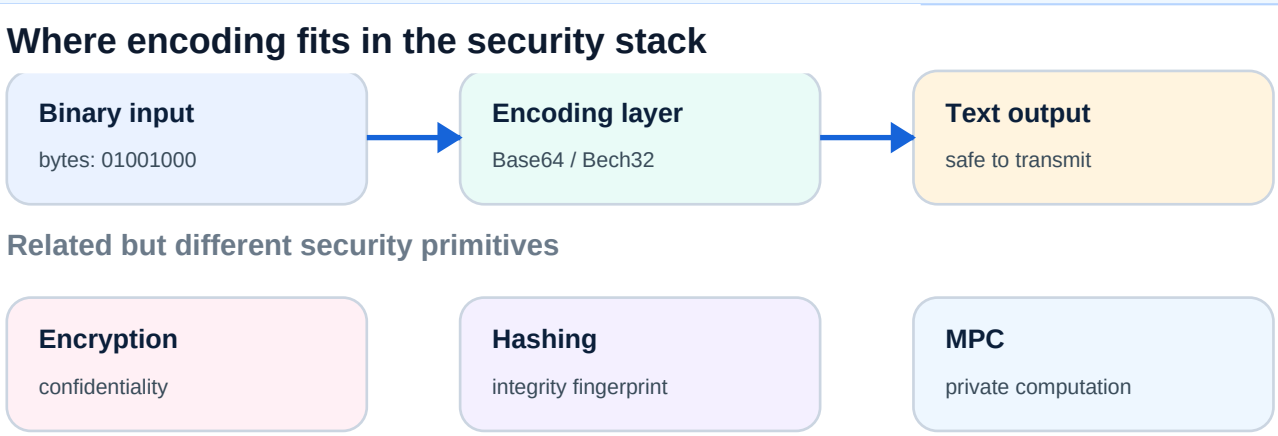


Figure 1. Encoding changes representation. Encryption, hashing, and MPC solve different security problems.

2. Base64: General-Purpose Binary-to-Text Encoding

Base64 maps binary input into a 64-character alphabet. Every 24 bits of input are split into four 6-bit values, and each value is represented by a printable character. When the input length is not divisible by 3 bytes, padding with the equals sign may be added to align the output to a multiple of four characters.

Correct example: The text `Hello` becomes `SGVsbG8=`. Decoding `SGVsbG8=` returns `Hello`. In URL-safe profiles, such as many JWT implementations, padding can be omitted when the decoder can infer the missing length.

Important correction: the major overhead of Base64 is not the padding characters. The main expansion comes from representing every 3 bytes as 4 text characters, which creates about 33 percent overhead before transport-layer compression.

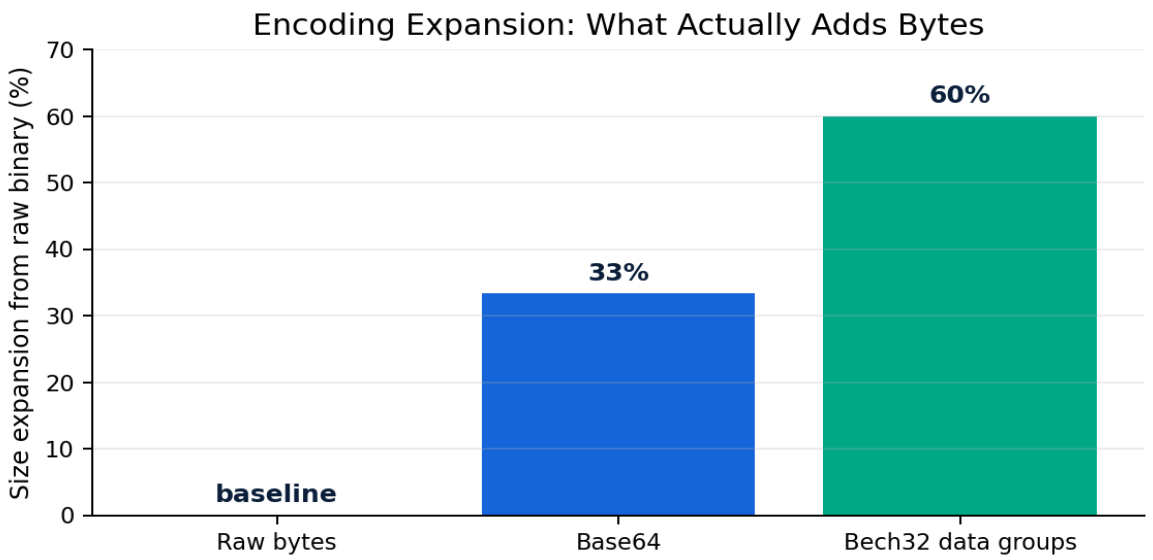


Figure 2. Base64 overhead comes primarily from the 3-to-4 byte expansion. Bech32-style 5-bit grouping has different trade-offs and is not a direct Base64 replacement.

Reading the chart: Bech32 is optimized around 5-bit groups and checksum behavior, not around being a smaller universal replacement for Base64. In user-facing wallet software, correctness and error detection can be more important than raw character density.

3. Bech32: Human-Friendly Bitcoin Address Encoding

Bech32 was introduced by BIP-173 for native SegWit Bitcoin addresses. Its design emphasizes reliable human transcription, lowercase readability, a restricted character set, and checksum-based error detection. A Bech32 string is not just arbitrary data encoded in Base32 form; it has a specific structure.

Bech32 address anatomy

HRP	Sep.	Data payload	Checksum
`bc`	`1`	`w508d6qejxtdg4y5r3zarvary0c5`	`ygt080`

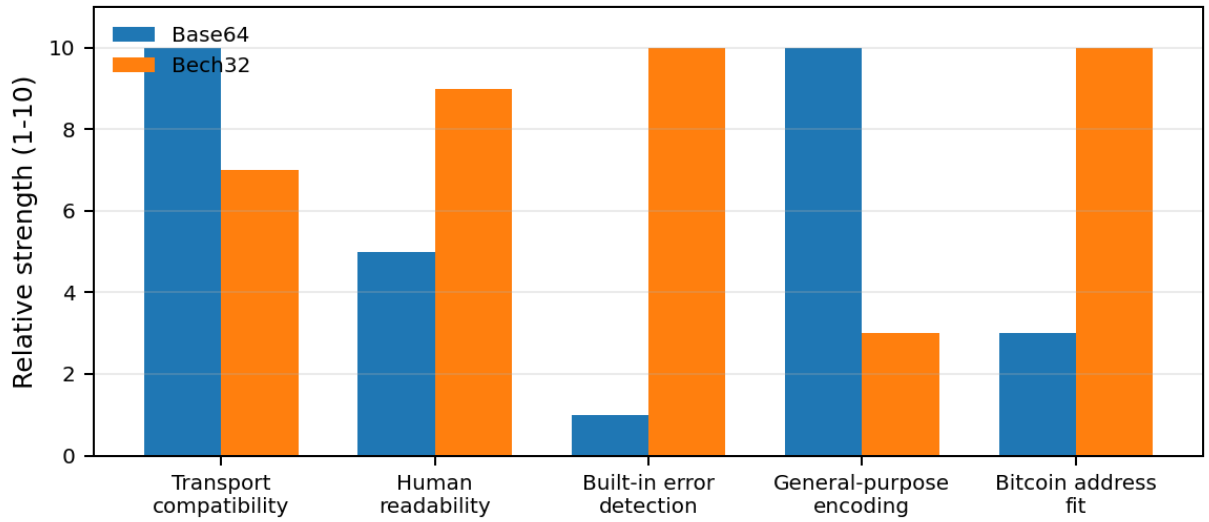
The checksum helps detect common typing and transcription mistakes before a transaction is sent.

Example: bc1qw508d6qejxtdg4y5r3zarvary0c5xw7kygt080

Figure 3. Bech32 combines a human-readable part, separator, data section, and checksum.

The earlier claim that a random phrase such as "SecureData123" directly decodes from an invented Bech32 address is inaccurate. Bech32 can encode data values, but Bitcoin addresses follow strict witness-program rules. Valid examples should use known BIP-173 test vectors or real address-generation logic.

4. Base64 vs Bech32: Practical Comparison



Property	Base64	Bech32
Primary goal	Transport binary data through text systems	Represent Bitcoin address data with readability and checksums
Standard	RFC 4648	BIP-173
Alphabet size	64 symbols	32 symbols
Padding	Commonly uses = padding; some profiles omit it	No conventional Base64-style padding
Error detection	No built-in checksum	Built-in checksum
Security role	Encoding only; not confidential	Address validation only; not confidential
Best use	APIs, email, JWT payload encoding, binary blobs in text	Bitcoin address representation and user-facing address entry

5. Correct Security Framing

A professional paper should separate representation from protection. The following taxonomy keeps the architecture accurate:

Technique	Question it answers	Example
Encoding	How can data be represented safely in a channel?	Base64, Bech32, hexadecimal
Encryption	How can data be hidden from unauthorized parties?	AES-GCM, ChaCha20-Poly1305
Hashing	How can data be fingerprinted for integrity?	SHA-256
Digital signature	How can a sender prove authenticity?	ECDSA, EdDSA
MPC	How can parties compute jointly without revealing private inputs?	Private set intersection, secure aggregation

This correction removes the weak direct comparison between Bech32 and MPC. Bech32 helps users and wallets detect address mistakes. MPC helps multiple parties compute on private inputs. Both can appear in blockchain ecosystems, but they sit at different layers of the stack.

6. Blockchain Implications

Bech32 improves address usability and validation. However, it is not accurate to say that Bech32 alone reduces transaction fees. Lower transaction costs are more closely associated with SegWit transaction structure and weight accounting. Bech32 is the user-facing address format that supports native SegWit addresses.

For protocol designers, the lesson is broader: address formats should reduce ambiguity, reject common human errors early, and make invalid states obvious before irreversible actions occur.

7. Design Guidance for Engineers

Use this decision guide when choosing a representation format:

Requirement	Recommended choice	Reason
Need to embed binary data in JSON, email, or text APIs	Base64 or Base64url	High compatibility and standard library support
Need a user-facing Bitcoin address	Bech32 / Bech32m as appropriate	Readable alphabet and checksum validation
Need confidentiality	Encryption, not encoding	Encoded data is reversible without a secret
Need tamper evidence	Hash, MAC, or signature	Encoding does not detect malicious changes
Need private joint computation	MPC protocol	Separates data privacy from representation format

8. Final Conclusion

Base64 and Bech32 are both important, but they are important for different reasons. Base64 is the practical workhorse for binary-to-text transport. Bech32 is a specialized, human-friendly address format with strong checksum properties for Bitcoin systems. A technically accurate paper should avoid claiming that encoding provides confidentiality or that Bech32 directly replaces cryptographic privacy protocols. The strongest design principle is simple: use encoding for representation, cryptography for security, and protocol-specific formats for user safety.

Key takeaway: professional secure protocol design is not about choosing the fanciest encoding. It is about matching each layer to the correct job.

Appendix A. Corrections Applied

Original weakness	Professional correction
Invented Bech32 example	Replaced with a known address-style example and explained Bech32 structure
Overstated Base64 padding cost	Clarified that the main overhead is 3-byte to 4-character expansion
Weak Bech32 vs MPC comparison	Separated encoding from cryptographic computation
Transaction fee overclaim	Explained that fee effects relate primarily to SegWit transaction weight
Encoding treated like security	Added a security taxonomy showing encoding, encryption, hashing, signatures, and MPC

References

[1] Josefsson, S. The Base16, Base32, and Base64 Data Encodings. RFC 4648, IETF, 2006.

[2] Wuille, P. Bitcoin Improvement Proposal 173: Base32 address format for native v0-16 witness outputs, 2017.

[3] Wuille, P. Bitcoin Improvement Proposal 350: Bech32m format for v1+ witness addresses, 2021.

[4] NIST. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.

[5] Goldreich, O. Foundations of Cryptography: Basic Applications. Cambridge University Press.

Efficient Encoding Methods for Secure Network Protocols

Pejman Haghighatnia

Technical Whitepaper - Professional Edition